

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
25 January 2001 (25.01.2001)

PCT

(10) International Publication Number
WO 01/06435 A1

- (51) International Patent Classification⁷: G06F 17/60 (71) Applicant (for all designated States except US): E-DIALOG, INC. [US/US]; 1646 Massachusetts Avenue, Lexington, MA 02420 (US).
- (21) International Application Number: PCT/US00/19403 (72) Inventor; and (75) Inventor/Applicant (for US only): ESTES, Anthony, D. [US/US]; 4 Westmoreland Avenue, Arlington, MA 02174 (US).
- (22) International Filing Date: 14 July 2000 (14.07.2000) (74) Agent: FEIGENBAUM, David, L.; Fish & Richardson P.C., 225 Franklin Street, Boston, MA 02110-2804 (US).
- (25) Filing Language: English (81) Designated States (national): AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.
- (26) Publication Language: English
- (30) Priority Data: 09/353,896 16 July 1999 (16.07.1999) US
- (63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application: US 09/353,896 (CON) Filed on 16 July 1999 (16.07.1999)

[Continued on next page]

(54) Title: DIRECT RESPONSE E-MAIL

WILLIAM HERP 10

FROM: HARVARD BUSINESS SCHOOL PUBLISHING
SENT: WEDNESDAY, DECEMBER 16, 1998 9:19 PM
TO: WILLIAM HERP
SUBJECT: == NEW INSIGHTS FROM HARVARD BUSINESS REVIEW

HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION
BOSTON, MASSACHUSETTS USA

THURSDAY, DECEMBER 17, 1998

DEAR WILLIAM HERP,

ON THURSDAY, DECEMBER 3RD, WE WROTE YOU REGARDING A SPECIAL OFFER ON THE HARVARD BUSINESS REVIEW PAPERBACK SERIES. SINCE WE HAVE NOT HEARD BACK, WE WANTED TO FOLLOW UP BEFORE THIS SPECIAL OFFER CLOSES. IF YOU ARE SIMPLY NOT INTERESTED, WE APOLOGIZE FOR THE INTRUSION. BELOW PLEASE FIND THE ORIGINAL OFFER IN ITS ENTIRETY.

THE HARVARD BUSINESS REVIEW PAPERBACK SERIES BRINGS YOU THE LATEST AND MOST SIGNIFICANT THINKING ON TODAY'S MOST PRESSING MANAGEMENT CHALLENGES. THESE INSIGHTFUL COLLECTIONS ARE THE DEFINITIVE RESOURCE FOR PROFESSIONALS.

EACH TITLE:
+ PROVIDES A BROAD UNDERSTANDING OF AN ISSUE
+ IS CLEARLY WRITTEN AND, IN MANY CASES, DRAWS UPON REAL COMPANY EXAMPLES
+ HELPS YOU CONSTRUCT A USEFUL CONCEPTUAL FRAMEWORK FOR DECISION-MAKING AND IMPLEMENTATION
+ CONTAINS EIGHT ARTICLES FROM HARVARD BUSINESS REVIEW

◇ EACH PAPERBACK IS \$19.95 PLUS SHIPPING AND HANDLING ◇

TO ORDER ONE OR MORE OF THESE PAPERBACKS, SIMPLY REPLY TO THIS MESSAGE AND NOTE THE LETTER (A-F) OF THE HARVARD BUSINESS REVIEW PAPERBACK YOU WOULD LIKE TO RECEIVE. PLEASE TYPE THE LETTERS IN THE 1ST LINE OF THE BODY OF YOUR REPLY E-MAIL. SHIPPING AND HANDLING CHARGES WILL BE APPLIED TO EACH ORDER.

YOUR CHOICES (DETAILED BELOW) ARE:

A: HARVARD BUSINESS REVIEW ON CHANGE PAPERBACK
B: HARVARD BUSINESS REVIEW ON KNOWLEDGE MANAGEMENT PAPERBACK
C: HARVARD BUSINESS REVIEW ON STRATEGIES FOR GROWTH PAPERBACK
D: HARVARD BUSINESS REVIEW ON MEASURING CORPORATE PERFORMANCE PAPERBACK
E: HARVARD BUSINESS REVIEW ON LEADERSHIP PAPERBACK
F: THE EXECUTIVE COLLECTION - ALL FIVE TITLES FOR \$89

OR IF YOU PREFER, CALL 1-800-668-6780 (617-496-1449 OUTSIDE THE U.S.)
MON. - FRI. 8 A.M. - 6 P.M. EST. PLEASE BE SURE TO MENTION PRIORITY CODE 3202.

(57) Abstract: An e-mail message [10] is analyzed to derive response information concerning a commercial transaction. Based on the derived information, commercial transaction data are automatically generated in a format that is usable to complete the transaction.

WO 01/06435 A1



(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *With international search report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

DIRECT RESPONSE E-MAILField of invention

5 This invention relates to direct response e-mail.

Background of the Invention

In direct response e-mail, a vendor, for example, can sell a product to a customer by sending an e-mail message to the customer that describes the product and its price. The customer can order the product by returning an e-mail (sometimes called a direct response e-mail) that gives appropriate order information. The vendor can confirm the order by a return e-mail. The order information returned by the customer can sometimes be determined automatically using software that analyses the customer's reply e-mail.

Summary of the Invention

In general, in one aspect of the invention, an e-mail message is analyzed to derive response information concerning a commercial transaction. Based on the derived information, commercial transaction data are automatically generated in a format that is usable to automatically complete the commercial transaction.

In general, in another aspect of the invention, an e-mail message is sent to a customer offering a product or service for sale. The e-mail message includes locations for response by the customer to indicate his intention to order the product or service. The customer returns an e-mail message that includes the response. Based on the received e-mail, order information is automatically generated in a format usable automatically by an order fulfillment system to cause the order to be filled.

In general, another aspect of the invention includes automatically identifying response information which

requires resolution of an issue with the source of the e-mail message and automatically managing an e-mail dialog with the source to resolve the issue.

In general, in another aspect, the invention
5 features automatically sorting e-mail messages, based on response information contained in the messages, into e-mail messages that can be processed automatically to generate commercial transactions, e-mail messages in which the response information is inadequate to permit generation of
10 commercial transactions, and e-mail messages that may be subjected to exception handling to yield information that is sufficient to generate commercial transactions.

In general, in another aspect, the invention
features automatically generating a confirmatory e-mail
15 message to the source of the e-mail message confirming that a commercial transaction has been or will be completed.

In general, in another aspect, the invention
features receiving inbound e-mail messages that result from
corresponding outbound e-mail messages associated with a
20 marketing program, the inbound messages containing response information, each of the outbound messages being associated with a distinct piece of the marketing program. The response information in each of the inbound messages is automatically associated with the corresponding distinct
25 piece of the marketing program.

In general, in another aspect, the invention
features automatically merging response information with
corresponding information in a database for use in
completing transactions.

30 In general, in another aspect, the invention
features identifying inbound e-mail messages that cannot be processed automatically to generate commercial transactions,

and using the database information to assist in exception handling of the identified inbound messages.

Other advantages and features will become apparent from the following description and from the claims.

5 Brief Description of the Drawing

Figures 1A through 1C and 2A through 2B show e-mail messages.

Figure 3 is a block diagram of a direct response e-mail system.

10 Description of the Preferred Embodiments
Outbound e-mail messages

The two e-mail messages shown in figures 1A through 1C and 2A through 2B are examples of outbound messages associated with commercial transactions.

15 The example message 10 shown in figure 1 offers Harvard Business Review products. Message 10 includes basic copy 12 that is similar to basic direct marketing copy of the kind that is commonly used in e-mail marketing. Message 10 also contains a section 14 giving instructions on how to
20 order the products.

Inbound e-mail messages

To take advantage of the offer shown in figure 1, the recipient creates a reply e-mail message (the direct response message) and types the letters of the items that he
25 wants to order in the first line of the body of the message. In other examples, the letters could be typed in the subject line or the last line of the body of the message. The user is also asked to correct and complete shipping and e-mail address information that has been merged into the outbound
30 e-mail message in a section 16. In section 16, each of the entries is bounded by brackets. Another section could

contain merged billing information, not shown. The person who replies to the e-mail (the customer) is meant to include the corrections or additions within the indicated brackets.

5 By allowing the recipient to take advantage of the offer simply by replying to the e-mail, rather than requiring the recipient to place an order by linking to a related web-site or to print the e-mail message and FAX it back, or to call an 800 number, a much higher return rate can be achieved. For conventional outbound e-mail messages
10 that require the recipient to click on an embedded URL to go to a web site, the returns may be on the order of several hundred percent on investment (the fee charged for delivering the outbound messages). By enabling the recipient to provide direct response e-mail messages as in
15 figure 1, the return on investment can be as high as several thousand percent.

Figures 2A and 2B illustrate a similar outbound e-mail message in which there is no choice of products but only a single offer to be accepted or rejected. To take
20 advantage of the offer, the recipient types "yes" in the subject line. In figure 2B, a shipping block 18 of the kind mentioned above is shown. (In this case, the shipping block contains no information because the shipping address is the same as the billing address.)

25 One reason for including differential billing and shipping blocks is to acquire information in the return e-mail message that is similar to information captured in orders placed on a related web site. In a system in which web-site orders generate fields that can be fed directly to
30 an automated order fulfillment process, it is useful to make the e-mail message information field-wise consistent to permit the information to be delivered automatically to the same order fulfillment process.

Exception processing

Processing the inbound e-mails (the ones with responses concerning commercial transactions from the recipients of the outbound e-mails) may require custom
5 interaction with the recipients. For example, the wording of the outbound messages may be confusing to the recipients.

As shown in figure 3, the system 40 enables the transactional e-mail message processor 42 to determine when a dialog with the recipient 44 is needed and then assists a
10 human service representative 46 to conduct an effective dialog 48. The dialog can be conducted on behalf of the vendor 50 but without involving the vendor. Alternatively, the vendor's fulfillment process 52 can be notified electronically 54 of interaction that may be required.
15 Easing the processing of responses that include customer orders is important because the orders typically come back quickly, e.g., within 36-48 hours, and in large volume. The ability to deal with questions that arise as a result of the contact from a customer service point of view keeps the
20 vendor's customer service organization from being overwhelmed by the responses that come back.

The ability to process exceptions without involving the customer service organization of the vendor is based partly on knowing how the outbound e-mail messages were
25 constructed. As a simple example, a recipient may ask an unnecessary question that could have been answered by reading the outbound e-mail message. The e-mail message processor can pull out the relevant portion of the message and send it back to the recipient to answer the question.

ProcOrder Process

The inbound e-mail messages 60 are batch processed by a script called ProcOrder 62. ProcOrder parses the elements of the inbound e-mail messages in accordance with the original set up and instructions of the outbound e-mail messages 64. ProcOrder determines if all of the items that are required for an order to be completely processed automatically appear in the inbound e-mail message. For example, the script would look for the ordering token, such as the word "yes" or a series of letters depending on whether it is a single or multiple offer. The script would also look for footer information in the e-mail message, including a code that identifies the given campaign and the given offer, as seen in block 66 of figure 2B. In that example, there are four components in the footer, but only two are represented because the other two are not required in this instance. The first element is a customer identifier 68, e.g. 861270. Then there is a space 70 between two pipes that would contain the list identifier if there were one. There may be multiple recipient lists for a given marketing campaign. In the example, there is only one list, and there is no list identifier. A list number 243 might refer to a list of people who made a purchase at the vendor's web site or who subscribed at the web-site for a listserv.

The third footer item could be a source of awareness code 72, e.g., 3275, which identifies a particular marketing campaign. For example, in the case of figure 2, the code could refer to a Benchmarking Three-part Video Series offer.

The last item in the footer, located between the final pipe and the first right bracket would be a flight identification code 74. A given campaign could have multiple flights of e-mail messages.

After looking for the footer information, the ProcOrder parser looks for fields in the billing and shipping address blocks that are required to complete the order. What is required may vary with the type of campaign
5 but typically the minimum requirements are a name and a physical address. If the information is not completely available in the response e-mail message, the script checks to see if it is available in the database 76. If not
10 available in either place, the script generates an exception entry for an exception list. The exception list is provided to a service representative 46 who can then act on it (without involving the vendor's customer service organization), e.g., by sending back an e-mail message asking for the shipping address.

15 If all required information is available, the script generates a fully fielded valid order in a format required by the fulfillment system of the vendor and adds it to a batch of valid orders 78 which are sent electronically to the fulfillment process.

20 Confirmation e-mail message

As a result of running the ProcOrder script, an e-mail message 80 is returned to each customer either to confirm an order or to request more information. In the latter case, a dialog ensues and is managed by software and
25 through an exception handling service as explained earlier. For example, the customer's response could say something like "sure, send"; or "send it and I'll take a look." Shortly thereafter the customer would receive a confirmation
30 "Thank you for your order; you can expect the CD-ROM in about seven business days. Please let us know if there is anything else we can do to help simply by replying to this e-mail."

One-click ordering

Another feature of the e-mail dialog with a customer involves simplifying and optimizing the presentation of content. In the examples of figures 1 and 2, the

5 information is presented in a simple text format. It is useful also to provide in-line HTML code in the outbound e-mail message in a manner similar to the one-click ordering that Amazon.com offers in a web-site context. In one-click ordering, the customer sets up an account by providing
10 credit card and shipping information. On subsequent visits to the web site, the customer can pick a product with one click, place an order, and have it shipped. A similar technique could be adapted to e-mail message interchange by embedding one-click ordering into e-mail.

15 An advantage of in-line HTML code is the opportunity for a much higher response rate because of the higher graphical contact and higher level of engagement normally achieved by a graphical message.

Template

20 The outbound e-mail messages are set up in a standard format using templates 90. The templates enable either a single-offer message or a multiple-offer message. Other templates are also possible, including one that embeds in-line HTML into the message as mentioned above, either for
25 the single-offer or multiple-offer cases.

In addition, a set-up tool 92 permits the parameters of a given campaign to be defined, including the source of awareness code, the flight identification code, the campaign identification code, and similar information. The set-up
30 tool also permits defining the tokens that are to be used in a given campaign (for example, the letters assigned to different products being offered). The set-up tool also allows a definition of the required fields that must appear

in a given campaign to enable automated generation of orders to an existing fulfillment system.

The set-up tool also provides a user interface that enables a vendor to help in entering the set-up information.

5 The result of applying the tool to the templates is a set of outbound message forms 94 that are ready for use.

Reporting tool

After the template is set up and the system is ready to launch a flight, address 108 and other information 110.
10 112 stored in the target list of customers is merged with the message forms, and the e-mail messages are automatically generated and sent by an outbound e-mail delivery engine 96. Customers then begin to respond. The ProcOrder script generates automatic orders to the fulfillment system and
15 exception information for additional processing.

A reporting tool 104 aggregates information about the responses for a given campaign according to source of awareness code and flight. The information is made available on-line to the vendor and can be used for a
20 variety of marketing purposes. The information could be generated as an Excel file attached to an e-mail, or as a paper-based report, or as an electronic file that is transferred on a batch basis.

Gathering additional information from database

25 There may be an intermediate step between the parsing engine's (ProcOrder) extraction of information from an e-mail message and the generation of the valid order. The intermediate step could be a querying process 112 to gather additional information from an existing database.
30 The additional information may not have been included in the outbound e-mail messages but may be needed to generate a valid order. For example, product codes 112 may be stored in the database but not included in the outbound e-mail

message. The letters entered by the customer can be mapped to the actual product codes by reference to the tables of the database based upon the source of awareness code.

5 The resulting valid order is a fully-fielded record that has the fields required by the client's order fulfillment system to process an order.

Exception treatment

Exception handling can be treated in different ways depending on the circumstances. For example, an exception
10 might occur when a customer responds from an e-mail client that does not quote the original text of the outbound e-mail message. The inbound e-mail message then has the customer's e-mail address, a subject line that says "yes", and the original subject line from the campaign, but does not have
15 the required information for the shipping address or the footer information. ProcOrder would kick that out as an exception, but the exception handling system would allow a response management representative 46, based on the e-mail address, to confirm, from the database 76, that all of the
20 required information is available. Use of the subject line allows the system to tie back to the appropriate campaign and to figure out who is ordering and what he is ordering. A valid order can be created without further interaction with the customer other than to send him a confirmation that
25 the system has been able to enter a valid order on his behalf.

The system thus recognizes that it is not likely to be possible to automate every interaction with the customer, but it may be possible to complete a dialog with essentially
30 all of the customers from whom inbound e-mail messages are received by automatically identifying messages that will require custom human handling and providing information and

tools that enable the human handlers to complete the exception transactions in an efficient manner.

Non-order response processing

Not every inbound e-mail message is an order. Non-order messages include undeliverable bounced messages to ad hoc customer service responses. Non-order inbound e-mail messages must be identified by the parsing engine.

Undeliverable e-mail messages 114 are automatically separated from the inbound e-mail stream and stored for offline handling by a human response handling professional, who operates a script on the files of undeliverable messages. The script classifies them as "soft" and "hard," parses e-mail addresses and footer data from the messages, matches the parsed records to the database, and flags appropriate records as "undeliverable".

Other non-order messages also are handled manually as explained earlier.

Vendor creation of e-mail campaigns.

A campaign creation tool 126 is provided to a vendor to enable simple entry of all information needed to create an e-mail campaign, including all the parameters, the text of the messages, and the tables of data needed in the database. The vendor delivers the campaign electronically to the transactional e-mail processor which then delivers the e-mail messages, receive the responses, processes all exceptions, and returns to the fulfillment system the vendor orders in a proper format.

A web-based vendor interface 128 enables on-line viewing by the vendor of the status of all campaigns, including the state of those that are in development and the results of those that are "live". The information is hosted by the transactional e-mail processor in part based on the

database 76. The interface also gives the vendor a mechanism to check text and other content into the database.

Alternatively, instead of automatically permitting the vendor to fully create a finished campaign, the vendor
5 may be enabled to download and check into the database a proposed campaign. Then an account executive of the e-mail handler process would review it and work with the vendor to complete it before it is finally queued for distribution.

Appendices A, B, and C contain more detailed
10 descriptions of aspects of implementations of the invention. Appendix D contains source code written of an example of the ProcOrder process.

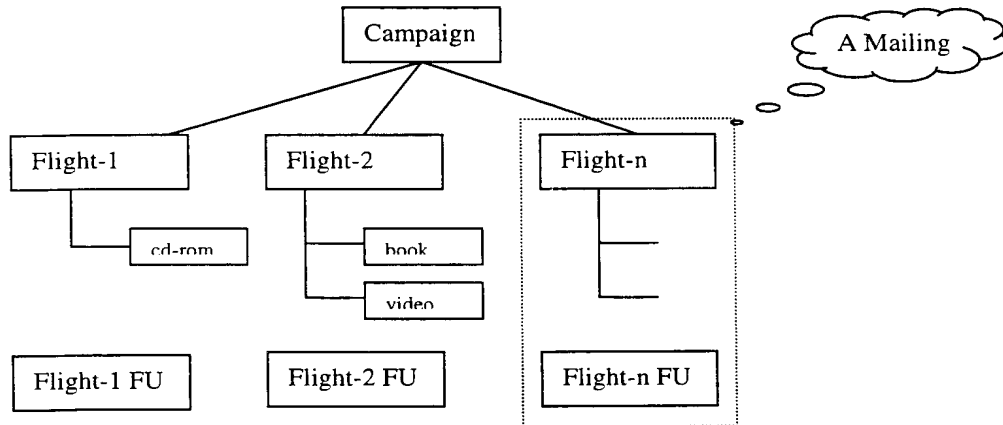
Other implementations are within the scope of the following claims.

15 What is claimed is:

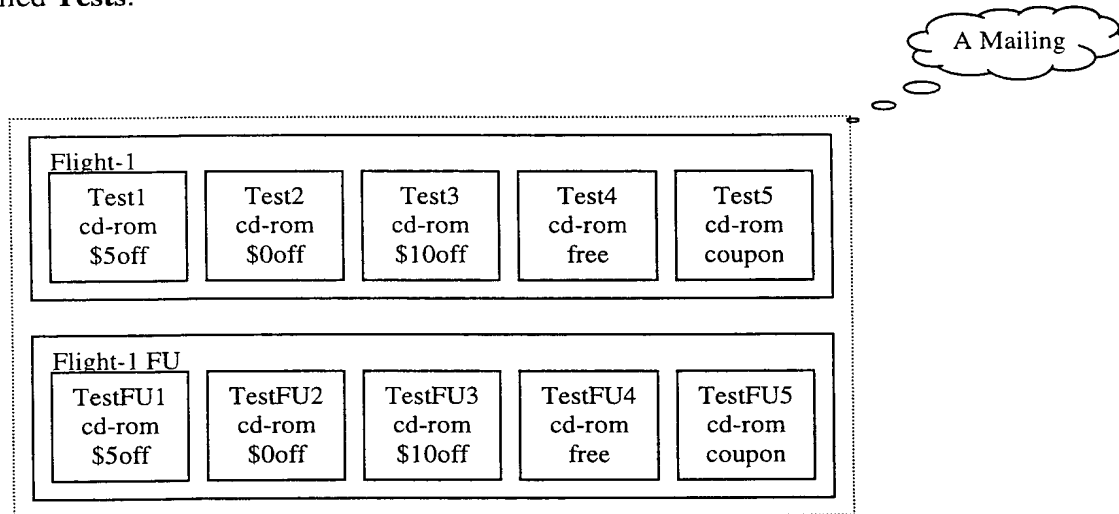
e-mail Protocols, Structures, Definitions & Cycles

MAIL CAMPAIGN

A mail campaign is defined in terms of **Flights**.



A flight contains one or more items that are being promoted but is not restricted to the manner in which the promotion is performed. A flight could be setup to promote a CD-ROM but has variances in the promotion of the item to the targeted mailing. These variances in a flight are called **Tests**.



In order to reference the entities above, each are assigned Ids as follows:

APPENDIX A

A Campaign

- Has a Campaign ID , **CID**
- Has 1 to n flights and flight follow-ups

A Mailing

- Has a Mailing ID, **MID**
- Has a Client Code (SOAC). The mailing is performed on behalf of the client.
- Has a Flight ID, **FID** and a Flight Follow-up ID, **FID**

A Flight

- Has a flight ID, **FID**
- Shares a **SOAC** with its Flight Follow-up
- Contains one or more Test Ids, **TID**
- Has 1 to m items for sale

An Item

- Has an Item ID **IID**
- Has an Item code (determined by the client)

To summarize, the following ID acronyms are used:

- | | | |
|-------|---|---|
| - MID | - | Member ID (ID of the person receiving the e-mail) |
| - CID | - | Campaign ID |
| - LID | - | maiLing ID |
| - FID | - | Flight ID |
| - TID | - | Test ID |
| - IID | - | Item ID |
| - SID | - | Style ID (ID of the style of the e-mail message) |

MAIL PROTOCOLS

SMTP

The *Simple Mail Transfer Protocol* (SMTP) is described in RFC 821 and is the way that two sites on the Internet exchange mail messages.

The commands are:

- HELO *domain*
- MAIL FROM: *username*
- RCPT TO: *username*
- DATA
- QUIT

Each command is terminated with a CR-LF pair. Replies start with a three-digit response code and continue with text designed to be read by users.

POP3

POP3 is the Post Office Protocol. If the site is always on the Internet, then mail would be sent with an SMTP-sender and received with an SMTP-receiver. However, in many cases it is not possible to maintain a permanent internet connection and in such cases, the Post Office Protocol is used to receive the inbound mail. POP3 allows mail to be stored on a machine that is always on the Internet and a receiving host connects to it, asks for any mail and disconnects.

The commands are:

- USER *username*
- PASS *password*
- STAT
- LIST *[message number]*
- RETR *message number*
- DEL *message number*
- LAST
- QUIT

THE ANATOMY OF AN OUTBOUND E-MAIL MESSAGE

An outbound email message has a predefined structure. The message is created by combining together a text block with an email address.

The structure of an outbound mail message consists of a Property, Address, Subject, Header, Body, Personal and Token.

Headers
Address
Subject
Salutation
Body
Personal
Token

Structure of an outbound e-mail message

a) Headers

The first few lines of an Outbound Mail message are called the Headers and have a defined format. This information is not normally displayed to the User by a Mail Client application and can only be viewed if the Client permits e.g., in Microsoft Exchange this information can be viewed by listing the properties of a mail message as follows:

```
Received: by mail.kersur.net (mbox peterk)
  (with Cubic Circle's cucipop (v1.31 1998/05/13) Fri Mar 19 20:59:00 1999)
X-From_: aestes@e-dialog.com Fri Mar 19 13:43:54 1999
Return-Path: <aestes@e-dialog.com>
Received: from montana.e-dialog.com (mail.e-dialog.com [207.31.244.2])
  by mail.kersur.net (8.9.1/8.9.1) with ESMTTP id NAA10659
  for <peterk@sytech.com>; Fri, 19 Mar 1999 13:43:53 -0500 (EST)
Received: by MONTANA with Internet Mail Service (5.5.2448.0)
  id <GZZPKKZ2>; Fri, 19 Mar 1999 13:43:57 -0500
Message-ID: <B15DC0490C8AD211BDFD004005A0C2CC47F10B@MONTANA>
From: Anthony Estes <aestes@e-dialog.com>
To: "'peterk@sytech.com'" <peterk@sytech.com>
Subject: FW: Warning: could not send message for past 4 hours
Date: Fri, 19 Mar 1999 13:43:54 -0500
MIME-Version: 1.0
X-Mailer: Internet Mail Service (5.5.2448.0)
Content-Type: multipart/mixed;
  boundary="-----_=_NextPart_000_01BE7238.71C0EB9A"
```

This message is in MIME format. Since your mail reader does not understand this format, some or all of this message may not be legible.

```
-----_=_NextPart_000_01BE7238.71C0EB9A
Content-Type: text/plain;
  charset="iso-8859-1"

-----_=_NextPart_000_01BE7238.71C0EB9A
Content-Type: application/octet-stream;
  name="ATT00547.TXT"
Content-Disposition: attachment;
  filename="ATT00547.TXT"
```

```

-----=_NextPart_000_01BE7238.71C0EB9A
Content-Type: message/rfc822

Message-ID: <B15DC0490C8AD211BDFD004005A0C2CC47EFF2@MONTANA>
From: Anthony Estes <aestes@e-dialog.com>
To: "Hillary Gaeth (E-mail)" <HGaeth@engage.com>
Subject: * Thurs, Fri ?
Date: Tue, 9 Mar 1999 08:04:20 -0500
MIME-Version: 1.0
X-Mailer: Internet Mail Service (5.5.2448.0)
Content-Type: text/plain

-----=_NextPart_000_01BE7238.71C0EB9A--

```

A header starts with a *field name* followed by a colon and the field body. The contents of the field body may be rigidly defined or free form.

The following headers are mandatory; that is, there must be a header with each of these names:

- Date
- From, or Sender and From
- To, or CC (carbon copy), BCC (blind carbon copy)

The following headers are optional:

- Return-path
- Received
- Reply-To
- Message-ID
- In-Reply-To
- References
- Keywords
- Subject
- Comments
- Encrypted

Some of these headers are obscure and rarely used. In addition, some mail clients generate their own extra headers. Many such extra header start with the characters "X-" because if extra mail headers are added to the RFC they will never start with these characters.

A header may be split over two lines according to the following rules:

- The split must be at a place where whitespace(blank or tabs) would normally occur; for example, not in the middle of a username or similar field.
- The continuation line must start with a space or tab.

Similar, a header can have extra white spaces almost everywhere except in the middle of a field.

Two headers that are given special attention in the context of this document are those that contain the address and subject.

b) Address

The address is a header that contains the email address of the person receiving the message.

c) Subject

This is a header that describes the subject of the mail. Contained in the subject is a **Subject Token** in the form of two characters. For example, for the mail subject '** A Special Offer for Selected Managers', the **Subject Token** is **

The **Subject Token** is chosen to identify the **Test ID** of the mail message.

d) Salutation

The salutation is a text block in the outbound mail message that is personalized to the person receiving the message. The intention is to provide a personalized greeting and an indication of the sender.

From: Harvard Business School Publishing <hbsp@e-care.com>
 To: 'esalas@proteisa.com.pe'
 Subject: ** A Special Offer for Selected Manager
 Date: Tuesday, July 14, 1998 6:47 PM

From the Desk of Laura Winig
 Harvard Business School Publishing Corporation
 Boston, Massachusetts

Tuesday, July 14, 1998

Ms. Elizabeth Salas
 5700 Collins Avenue Apt 6h
 Miami Beach, FL 33140-2308

Dear Ms. Salas:

The information in italic is personalized and is combined with the remainder of the message at the time of dispatch.

e) Body

The body contains the main core of the message. The format and layout is fixed and not personalized for a particular test.

Simply type "YES" in your reply to this e-mail to take "Virtual Work: Real Results" for a No-Obligation Test Drive!

=====

Do you work with colleagues and clients in multiple locations? Communicate more by email and conference calls than through meetings? Find your "office" is wherever you are at any given moment? Then you're working "virtually" and you know that working effectively without proximity is essential in today's workplace. And you know it's not as easy as it looks.

Here at the publishing arm of Harvard Business School, we've combined extensive research and real-life examples to create "Virtual Work: Real Results"-a dynamic multimedia program that can improve your effectiveness working "virtually."

=====

Use this engaging tool and you'll understand:

- * the dynamics and politics of working virtually-
and how to handle tough situations when you can't be face-to-face;
- * how to most effectively use email, video conferencing,
voice conferencing-and how to overcome fear of technology;
- * ways to build relationships and trust without "human touch"-

and how a virtual team can work efficiently and seamlessly.

 We bring you tricks of the trade from the experts in working virtually.
 Then, you gain confidence using these techniques through an interactive case study where you lead
 a virtual team through a project. You'll make decisions that determine the success of its
 efforts-all in a realistic, but no-risk, environment.

Overcome the isolation and conflicting loyalties that are inherent in working in a virtual
 environment-and get ready for success with "Virtual Work: Real Results."

 Take "Virtual Work: Real Results" for a No-Obligation Test Drive.

Simply reply to this email and we'll send you the program with our compliments. We're confident
 you'll find you're working more effectively in the virtual world. After 14 days, we'll send you
 an invoice for just \$295 (single user license).

But remember, if you're not entirely pleased with the program, simply call us and we'll arrange
 to pick it up. You will owe nothing.

Sincerely,

Laura Winig
 Director

In the above example the response with a "YES" is sufficient to indicate the purchase of the
 single product on offer. In a multi-product offer, the items would be listed and associated with a
 letter. For this type of mailing, the responder would list the letters in the response.

A mailing may have a follow up 'reminder' flight. This reminder would not go to respondents of
 the original flight. For example:

On Wednesday, July 1, I sent you a special offer on "Virtual Work: Real Results": a new
 interactive CD-ROM from Harvard Business School Publishing.

Since I haven't heard back from you, I wanted to send you a reminder Before the offer expires.

If you are simply not interested, I apologize for the intrusion.

----- BELOW IS A REPRINT OF THIS SPECIAL OFFER -----
 Simply type "YES" in your reply to this e-mail to take "Virtual Work: Real Results" for a No-
 Obligation Test Drive!

 Do you work with colleagues and clients in multiple locations? Communicate more by email and
 conference calls than through meetings? Find your "office" is wherever you are at any given
 moment? Then you're working "virtually" and you know that working

...
 ...
 ...

confident you'll find you're working more effectively in the virtual world. After 14 days, we'll
 send you an invoice for just \$295 (single user license).

But remember, if you're not entirely pleased with the program, simply call us and we'll arrange
 to pick it up. You will owe nothing.

Sincerely,

Laura Winig
 Director

f) Personal

The personal is added to the outbound message, after the mail body. The purpose of the text
 block is to request personal details from the respondent. The information requested is presented

in two columns, the first indicating the type of information required and the second as a place for the reply to be entered (in [..]). For example,

```
FIRST NAME:      []
LAST NAME:       []
TITLE:          []
COMPANY:         []
DEPARTMENT:     []
ADDRESS1:        []
ADDRESS2:        []
ADDRESS3:        []
CITY:           []
PROVINCE/STATE:  []
POSTAL/ZIP CODE  []
COUNTRY:         []
PHONE:          []
FAX:            []
EMAIL:          []
```

These details are to determine the shipping and billing information.

e) Token

The token follows the Mail Body (or Mail Personal if applicable) and contains information about the mailing and also the addressee e.g., [[878119|2815|1]]

The format of a Mail Token is

[[MID | TID | SID]]

where

- MID is an membership ID assigned to the person receiving the outbound message
- TID is the test ID assigned to the outbound message. This would be related to the Subject Token.
- SID is the style ID assigned to the style of the mailing i.e., single product or multi-product.

These three pieces of information uniquely identify the receiver of the outbound message and also the information they received. Hence, the processing of a response is greatly simplified if the reply returns the mail token.

In summary, an outbound message contains generic information that is the same in all the mailings of a test, and also personalized:

Generic
Subject
Body

Personalized
Header
Address
Salutation
Personal
Token

THE ANATOMY OF AN INBOUND E-MAIL MESSAGE

An inbound email message has a predefined structure. However, the structure may not be 'structured' sufficiently for it to be automatically processed.

Header
Address
Subject
Response

a) Headers

The first few lines of an Inbound Mail message are Headers and have a defined format. This information is similar in format to that of the outbound message. If the response is produced by replying to an outbound message, (instead of creating it from scratch) then it is probable that the headers of the outbound will be in the inbound mail.

b) Address

The address is a header that contains the email address of the respondent.

c) Subject

This is a header that describes the subject of the mail. This is free-form text and no assumptions can be on its structure. It may be the subject that was used in the outbound mail.

d) Response

The response is a text block containing the message from the respondent. It should not be assumed that the response has any structure since a responder has the freedom to write a reply in "free format" and is not forced to a guideline. This creates a number of problems for the processing of an inbound response since rigid rules cannot be applied.

The points that can be noted about a response are that:

- All responses will contain headers.
- All responses will contain the responders email address.
- All responses will contain a subject. The textual content of the subject cannot be assumed since it can be freely edited and so may not resemble the content of the outbound. If a subject contains a Subject Token then it should correspond to the TID.
- If the response contains the Mail Token then the MID, TID and SID will be available and consequently it will be clear on the approach that should be taken in processing the response i.e., the handling of a single product 'YES' versus a multi-product 'ADG'.

AN E-MAIL LIFE CYCLE

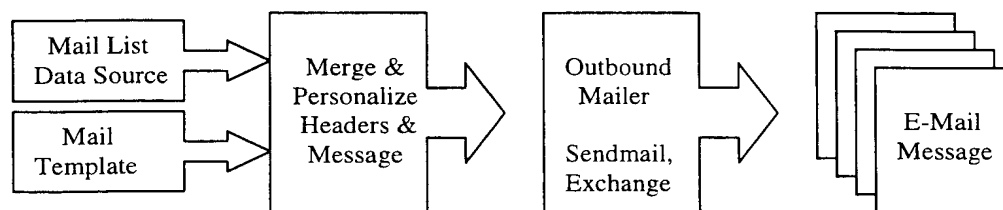
OUTBOUND

a) Create an outbound message

An outbound email message consists of texts block that contains information and one or many questions or choices. The text block of an outbound mail message is sometimes referred to as a **Mail Template**.

b) Deploy the outbound message

The message can be deployed to anyone for whom an e-mail address is available. The address information in a **Mail List** (e.g., database, listserve) is merged with the **Mail Template** and sent. In some cases, the X-Headers are also personalized to reflect the purpose of the outbound mail message.



INBOUND

a) Receive a Reply

The content of a reply can have various forms

- The reply responds to the outbound message e.g., an order.
- The reply requests to unsubscribe from future mailings. The reply contains an indication that the respondent does not wish to receive any further mailings e.g., UNSUB, UNSUBSCRIBE, UNJOIN and REMOVE.
- The reply requests customer service and ad hoc questions.
- The reply indicates a change in personal details or to be added to the mailing list.

There is also the possibility that the outbound message did not reach its final destination and that it bounced back. There are two categories of bounces, hard bounces and soft bounces.

- A hard bounce notification indicates outright failure. For a hard bounce, the subject would contain a message of the following form:
Returned mail: Host unknown (Name server: ssoofftteecchh.com: host not found)

The sender of a hard bounce is usually specific such as:

Mail Delivery System [MAILER-DAEMON]

A hard bounce can also be detected in the response X-Header:

```
Received: by mail.kersur.net (mbox peterk)
  (with Cubic Circle's cucipop (v1.31 1998/05/13) Fri Mar 19 21:04:39 1999)
X-From_: MAILER-DAEMON Fri Mar 19 21:04:36 1999
Return-Path: <MAILER-DAEMON>
Received: from localhost (localhost)
  by mail.kersur.net (8.9.1/8.9.1) with internal id VAA20320;
  Fri, 19 Mar 1999 21:04:36 -0500 (EST)
Date: Fri, 19 Mar 1999 21:04:36 -0500 (EST)
From: Mail Delivery Subsystem <MAILER-DAEMON>
Message-Id: <199903210204.VAA20320@mail.kersur.net>
To: <peterk@sytech.com>
MIME-Version: 1.0
Content-Type: multipart/report; report-type=delivery-status;
  boundary="VAA20320.921981876/mail.kersur.net"
Subject: Returned mail: Host unknown (Name server: ssoofftteecchh.com: host not found)
Auto-Submitted: auto-generated (failure)
```

The response body can also contain failure information:

The original message was received at Fri, 19 Mar 1999 21:04:35 -0500 (EST)
from dialup111.kersur.net [207.180.95.76]

```
----- The following addresses had permanent fatal errors -----
<JohnSmith@SSOOFFTTEECCHH.com>
```

```
----- Transcript of session follows -----
550 <JohnSmith@SSOOFFTTEECCHH.com>... Host unknown (Name server: ssoofftteecchh.com: host not
found)
```

- Softbounces can be of three types:

- *NonDeliveryNotification* occurs when a given message has not been delivered yet but will continue to try and deliver for a further specified period of time. This state can be detected in the response subject:
FW: Warning: could not send message for past 4 hours

The response can also be detected in the response body:

```
*****
**      THIS IS A WARNING MESSAGE ONLY      **
** YOU DO NOT NEED TO RESEND YOUR MESSAGE **
*****
The original message was received at Tue, 9 Mar 1999 08:00:18 -0500 (EST)
from mail.e-dialog.com [207.31.244.2]
```

```
----- The following addresses had transient non-fatal errors -----
hgaeth@andexc01.cmgi.com
(expanded from: <HGaeth@engage.com>)
```

```
----- Transcript of session follows -----
hgaeth@andexc01.cmgi.com... Deferred: Connection refused by
andexc01.cmgi.com.
Warning: message still undelivered after 4 hours
Will keep trying until message is 3 days old
```

- *AutoResponders* are notifications which actually indicate delivery but are sent by mail agents to indicate that the user will not be able to respond immediately (possibly on vacation) but the sender should expect a response when they return.
- *Unknown*

b) Process the reply

Automatic processing of an e-mail response is defined as the ability to determine accurately the requirements of the reply by using text inspection and search rules.

To automatically process an e-mail response, there are a number of criteria that need to be satisfied by the content of the reply. The main three criteria are:

- ci) Who is the Respondent
- cii) What outbound message is the response to
- ciii) What does the Respondent require

The exception to the above are hard and soft bounces that are identified by other e-mail properties.

Since all legitimate mail responses will contain the e-mail address of the Sender this can be regarded as a base information for all responses. This information fulfills criteria ci) but alone is not sufficient for a response to be processed automatically.

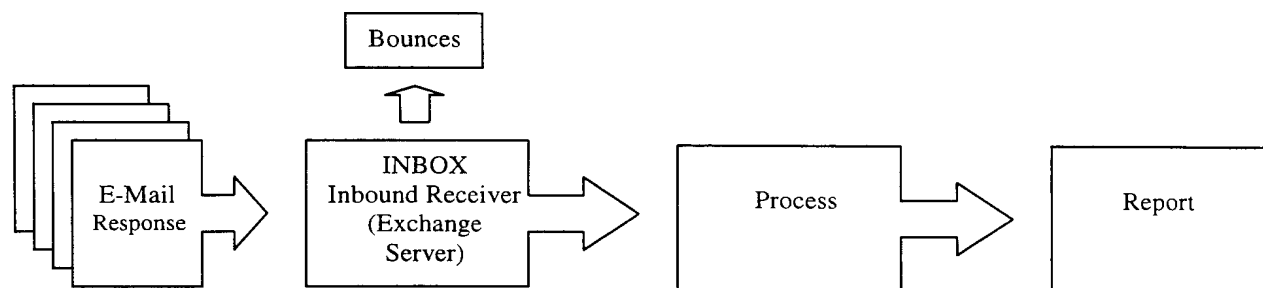
Criteria cii) can be satisfied by the Subject Token, provided the respondent has not altered it. Note that the Style ID is implied if criteria cii) is satisfied.

Another source of information is the Mail Token. For a reply that contains a Mail Token, both criteria ci) and cii) can be determined.

But how is criteria ciii) satisfied? There is no simple solution to this question since the answer lies in the body of the reply, which is in 'free form'. The only method available that can determine the requirements of the respondent is to parse the email reply based on the SID.

c) Produce Reports

After determining the requirement of the inbound e-mail, database entries and reports can be produced.



Design Specification for QuickReply

QUICKREPLY

The QuickReply application is used to process e-mail responses to mailing campaigns. The building and sending of the e-mail messages is not within the scope of QuickReply.

The test within a campaign's flight provides the outbound e-mail message. The message will contain both generic and personalized information and will normally require a response from the person the message is sent to.

Responses to a test are received in the Inbox folder of the mailbox. The design objective of QuickReply is to read the responses from the Inbox, and decide from its content the requirements of the response. On determining (or not determining) the requirement, the response is filed to pre-configured sub-folders and the appropriate tracking database is up-dated.

QuickReply will generate diagnostic information that will support the reasoning on how it reached its decision.

The main data storage area of QuickReply is Microsoft Access relational database. Appropriate records will be added to this database as the response processing is performed. In addition, it is likely there will be responses that cannot be automatically processed and, in such cases, Customer Service will use QuickReply in manual mode. Manual mode will also add records to the appropriate tables in the database.

The information in the database will be used to generate reports.

MAILBOX ORGANIZATION

Mailings are performed on behalf of a Client. Consequently, all e-mail transfers for a particular Client will be performed within the Clients mailbox.

The organization of the Clients mailbox includes an Inbox for the responses and a pre-configured number of sub-folders where the responses are filed.

Mailbox - <Client Reference e.g., HBSP>

Campaign CID

Flight FID

AC

YYMMDDhh

AddChange

YYMMDDhh

BouncesHard

YYMMDDhh

BouncesSoft

YYMMDDhh

Master

Order

YYMMDDhh

Unclear

YYMMDDhh

Unsubscribe

YYMMDDhh

Inbox

YYMMDDhh

In the above, YYMMDDhh is a directory named after the year, month, day and hour. This format reflects the date and time the entries were placed in that folder.

Each test flight will have a number of responses associated with it. When QuickReply categorizes a response, it will be automatically moved from the Inbox to the appropriate sub-folder.

- **AC** Responses for customer service to process manually
- **AddChange** Responses that contain personal detail changes
- **BouncesHard** Hard bounces
- **BouncesSoft** Soft bounces
- **Master** Master version of the outbound mail
- **Order** Responses with an order
- **Unclear** Responses that are unclear as to how they should be processed
- **Unsubscribe** Response to unsubscribe

E-MAIL STYLES

The Style ID (SID) of a response indicates the approach that should be taken by QuickReply during processing.

The SID is a two-digit numerical value

XY

where:

- X=1 indicates the outbound Personal block is included in the outbound message.
- Y=message layout style described below.

The following two layout styles are supported:

a) Single Product Offer – layout style=1

This style offers one product. The respondent is asked to reply “YES” if they are interested.

b) Multi-Product Offer – layout style=2

The mailing information is based on information in a relational database. This style offers a number of products the respondent can choose from. For example, the outbound message could contain:

Your choices (detailed below) are:

- A: The work of Leadership Audio
- B: Leading Your People Audio
- C: Leading Change Successfully Audio
- D: Overcoming Resistance to Change Audio
- E: Gaining Competitive Advantage Audio

The respondent makes a choice and responds by stating their requirement in the free format e.g., “please send me ABE”.

The number of items can vary as well as the choice symbol i.e., 1,2,3, could be used instead of A,B,C.

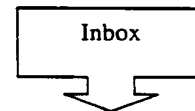
PROCESSING INBOUND MAIL AND REPORTING

The general processing and reporting function is described below. The general flow in processing responses is to determine the membership ID (MID), then the test ID (TID) and style ID (SID) and then perform an examination of the response to determine the requirement.

The processing criteria can be summarized as follows:

- ci) who is the Respondent?
- cii) what is the response to?
- ciii) what does the Respondent require?

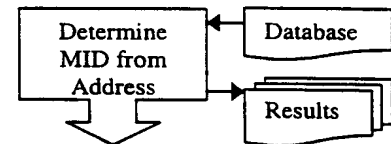
All inbound responses are targeted to the Inbox of the Client's Mailbox. On a periodic basis, the responses in the Inbox will be processed.



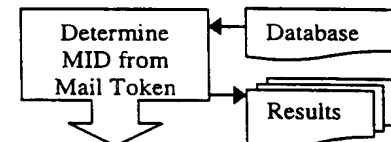
ci) Who is the respondent? – Determine the MID

There are several ways in which the Membership Information can be determined. The importance of the MID is that it could provide a handle to the Member's personal information.

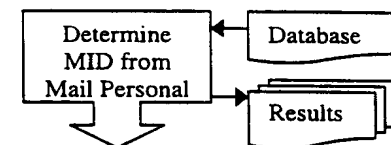
a) Extract the Response Address from the response. Determine if a unique record exists for the Address. If a unique address exists then the MID is determined. If the record is not unique then examine the mail token.



b) If the Mail Token exists then the MID is determined. Use the MID to confirm that the respondent's e-mail address is the same as that in the database. If the addresses are different then report that this was the case but continue to process.



c) If the SID indicates that the Personal block was included in the outbound, then extract the Member details from the Mail Personal. The extraction assumes that the response information is in square brackets '['']. The field definitions are defined in the database together with an indication of those that have to be completed by the responder. If Mail Personal is not complete (the fields are indicated in the database), move the response to Unclear and report.

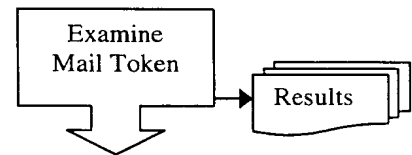


d) If the MID cannot be determined, move the response to Unclear and report.

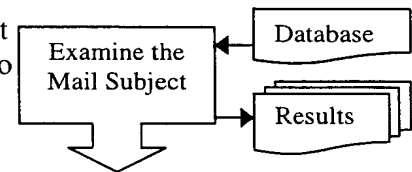
cii) What is the response to? – Determine the TID and SID

The next stage in response processing is to determine what the response is to. This can be determined from the Test ID.

a) If the Mail Token exists then extract the TID and SID. If the Mail token is absent, report but continue the determination of the TID.



b) Determine the TID from the database based on the Mail Subject of the response. If the look-up is successful, then the SID will also be determined.

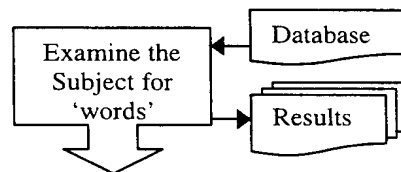


c) At this stage the TID and SID should be known. If not, move the response to Unclear and report.

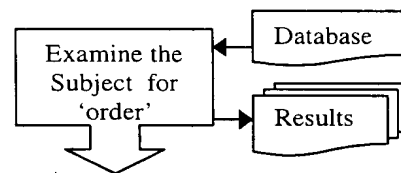
ciii) What does the Respondent require?

With the MID, TID and SID determined, the response message(s) are still of no value until it is established what the respondent requires. The obvious approach to this problem is to compare the response with the original and try to make some sense of the differences.

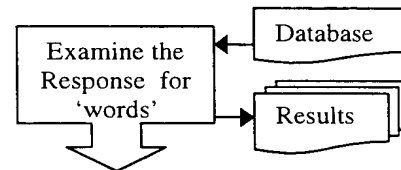
a) If the Mail Subject did not contain the Subject Token, examine for words that appear in the word action table. Any bounced responses are moved to the Bounce sub-folders. Any unsubscribe requests are moved to the Unsub sub-folder.



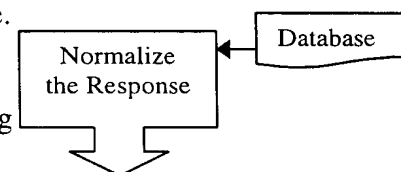
b) If the SID indicates a single item promotion, examine the Mail Subject for words that appear in the word action table. Any order responses are moved to the Orders sub-folders.



c) Examine the response for words that appear in the word action table. Any bounced responses are moved to the Bounce sub-folders. Any unsubscribe requests are moved to the Unsub sub-folder.

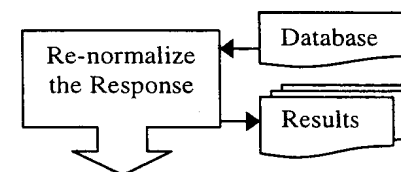


d) Remove all the text of the outbound message from the response.

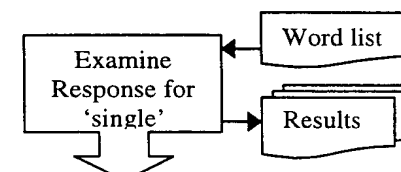


e) Normalize the response. Remove characters from the beginning of each line. The list of characters to be removed are in the word replacement table with ModeID=LineStart.

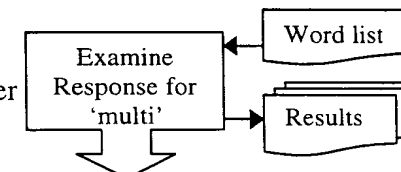
f) Replace phases from the normalized response. The phases that are replaced are given in the word replacement table with ModeID=PhaseReplace.



g) If the SID indicates that there is only a single item on promotion, examine the normalized response for words that indicate an order of a single product item. The words are in the word action table with the ActionID=SingleOrder. If order



h) If the SID indicates that there are multi-items on promotion, examine the normalized response for phrases that indicate an order of multi-items. The words are in the word action table with the ActionID=MultiOrder. In the case of multi-products, the



information the StartOption and EndOption from the bindings table are used to verify that item(s) ordered are in the correct choice range.

i) Any order requests are moved to the Orders sub-folder.

j) If the purpose of the response cannot be determined, then move the response to the Unclear sub-folder.

DATABASE ORGANIZATION

QuickReply uses two types of databases that are stored in Microsoft Access. The first database is part of the QuickReply application and contains high-level information on the campaigns, flights and tests. The second database is specific to a particular Client Test.

a) QuickReply Database

The QuickReply database contains the information that binds campaigns, flights and tests.

tblTestBindings	- contain Test bindings
Test ID	unique
Client ID	
Client Test Database	database name used for the test information
Subject	subject text of the outbound test
Style ID	the style of the outbound message
StartOption	character of the first item in the test
EndOption	character of the last item in the test
Campaign ID	
Mailing ID	
Flight ID	
tblOutboundPersonal	- contains personal details that appear in a test
Test ID	unique
Question	prompt displayed in personal block e.g., First Name
AnswerLen	the maximum length of the answer
AnswerReq?	flag to indicate that an answer must be given
tblWordActions	- contains words that convey an action
Word	unique
Action ID	action id for this word e.g., 'bounce'
tblWordReplacement	- contains phrases and their replacement
Phase	unique
Replacement	replacement to the phrase
Mode ID	type of replacement

b) Client Test Database

tblMembers	- contains member details
Member ID	unique
Email	
Prefix	
First	
Middle	

Last
Suffix
Title
Company
BillAddress1 ShipAddress1
BillAddress2 ShipAddress2
BillAddress3 ShipAddress3
BillCity ShipCity
BillState ShipState
BillZip ShipZip
BillCountry ShipCountry
Tel
Ext
Fax

EDRMA

Introduction

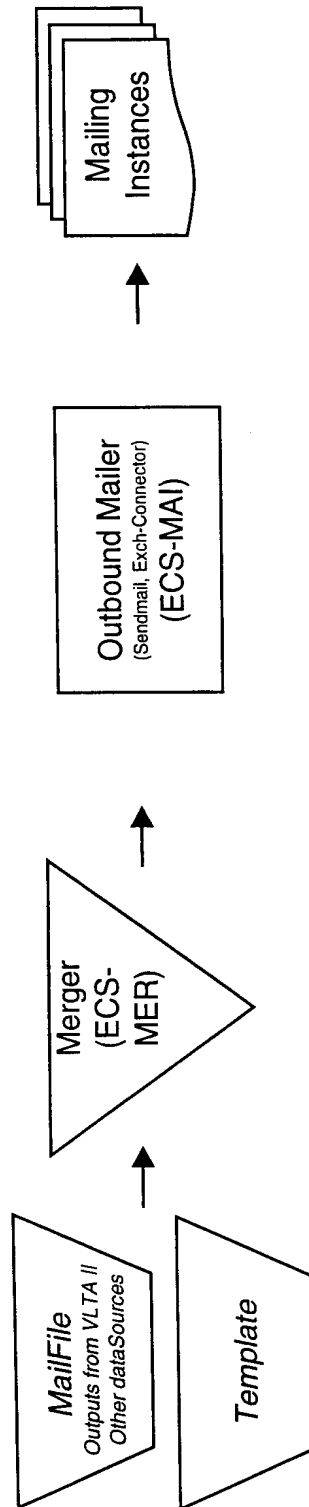
This document is intended to provide a schematic synopsis of E-Dialog Response Management Architecture (EDRMA).

Thereby it is hoped that the design of the Verbind E-Mail Channel Server (ECS) can best accommodate EDRMA's input requirements, and so EDRMA's output requirements may be adjusted to match those of Verbind LifeTime's architecture (VLTA)

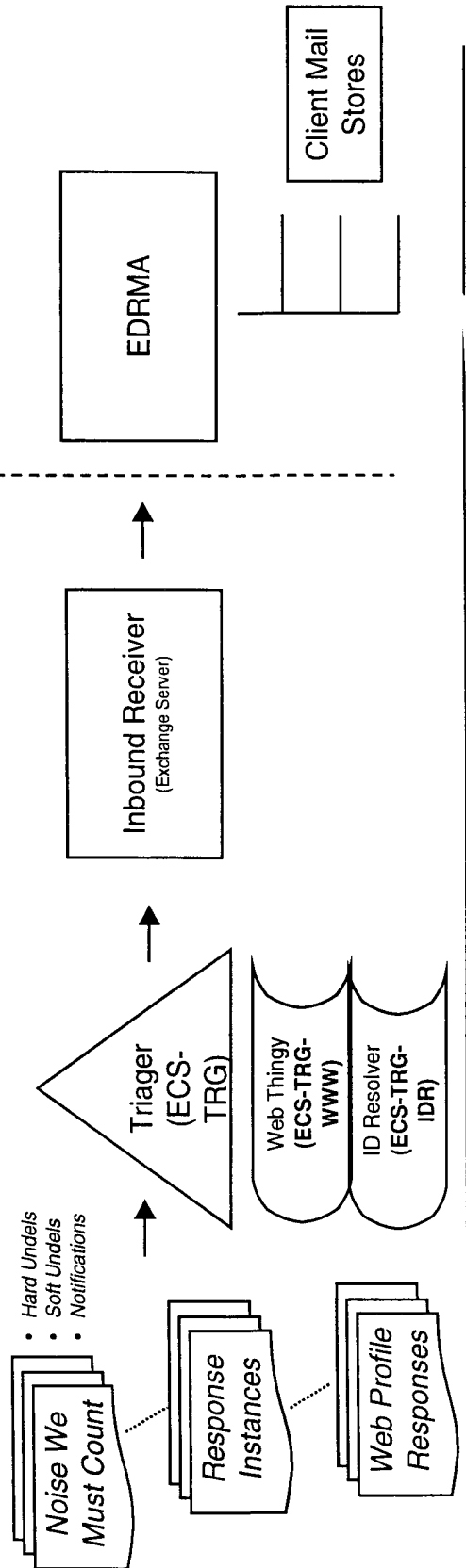
“First, some acronyms...” - *Anonymous*

EDRMA **Relevant Modules - Top [0.1]** **Where EDRMA Fits w/ECS**

OUTBOUND



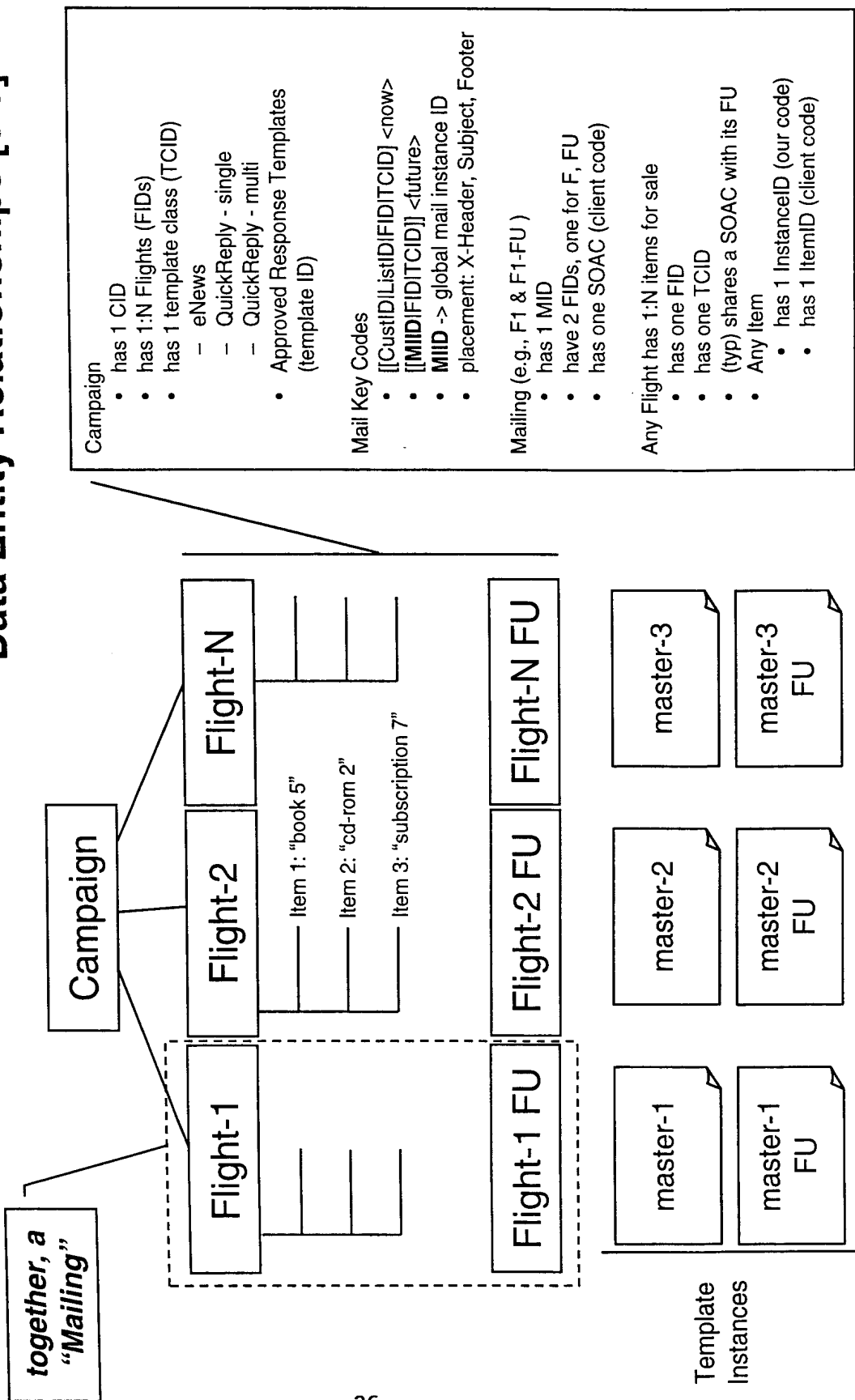
INBOUND



Bold modules indicate ECS modules

EDRMA Model

Data Entity Relationships [0.5]



EDRMA

Mailbox

Canonical Data Structures - Exchange Server Data Store [0.2]

Mailbox - <ClientFullName> (not alias: "hbsp")

<CampaignName>[<CID>]

- ApprovedResponseTemplates <templateID>

- <FlightName>[<FID>]

- AC (i.e., "Advocacy Care")

- + <date: YYYYMMDDhhmm>

- ADDCHANGE

- + <date: YYYYMMDDhhmm>

- HARDBOUNCES (i.e., hard undelivered... "jsmith@foo.com is not a valid addressee")

- + <date: YYYYMMDDhhmm>

- INBOX (monolithic, this Mailbox)

- MASTER (contains master template, this FID)

- ORDERS

- + <date: YYYYMMDDhhmm>

- » [[custID|ListID|FlightID|TCID]] << footer tags (x-header, subject)

- SOFTBOUNCES

- + DELIVERYNOTIFICATIONS

- » <date: YYYYMMDDhhmm>

- + AUTORESPONDERS

- » <date: YYYYMMDDhhmm>

- + UNKNOWN

- » <date: YYYYMMDDhhmm>

- UNCLEAR

- + <date: YYYYMMDDhhmm>

- UNSUBS

- + <date: YYYYMMDDhhmm>

EDRMA **Selected Applications**

VBA.PKinboxInspector
 interface: GUI / VBA
 data: MAPI
 inbox sorting, folder management application

VBA.PKresponseProcessor
 interface: GUI / VBA
 data: MAPI
 response review and report preparation application

PL.AEprocOrder()
 interface: commandline
 data: ADO 2.0
 process raw "order e-mails" by TCID
 uses of file rule sets for document preprocessing and data element parsing tied to CID; other rules, hardcoded
 generates:
 Acceptable output for review, annotated
 Exceptions, annotated
 Truncated BODYs, annotated
 Raw fields output , annotated
 Raw fields exceptions , annotated
 Rule-eval log (exhaustive)

PL.AEprocBatPrep()
 interface: commandline
 data: ADO 2.0
 takes selected output from PL.AEprocOrder() and transforms to a batch transfer specification via a field map and transform rules of

PL.AEprocUpdateBatVerbind()
 TBD

PL.AEscrubNormalCanon()
 << not used on the response side >>
 interface: commandline
 data: file handles
 scrubber, normalizer, canonicalizer
 also generates scrambled (non-predictable) row ids and flags AOLs

EDRMA

Process Stages [0.5]

Preliminary ECS hooks anticipated

Preprocess Stage

- inspect Inbox using VBA.PKInboxInspector
 - auto-creates folder structure (previous slide)
 - facilitates auto-sort of items into ORDERS, UNSUBS, UNDELS, HARDS, SOFTS, ADDCHANGE, etc.
 - facilitates manu-sort of items into AC, and exception
- report preparation by TCID via AEprocOrder()
 - produces: OUT_report, EXC_report && EXC_BODY diagnostic report

Processing Stage using VBA.PKresponseProcessor

- for each EXC_
 - inspect, correct, reconcile (via Mailing Table lookups) and commit
- for each OUT_
 - inspect and commit to report (or not)
- Acceptable UNION of EXC_ && OUT_ -> Reporting, Update Stages { AEprocBatPrep() || AEprocUpdateBatVerbind() }
- Hard exceptions are tagged as such and become Followup RFC's to get additional (critical information)
- FUP (FUPID -> RFC) tags affect routing; tie-back to HARD exception row in this EXC_ report for closure

Response Follow-up Confirmation Stage

- for each non-BOUNCE (ORDERS, UNSUBS, etc.), respond to respondents with an appropriate "confirmation of receipt/action taken" message
- using relevant response template store (**templateID**)
- this class of response likewise tagged for routing (FUPID -> confirm)

Reporting Stage

- produce order report
 - deliver via fax
 - post to client private web application
 - deliver as formatted batch

Update Stage

- deliver formatted batch to {Verbind, Database} -> sp_???, SQL Executive
-

F:\Development\PR\PR1_hbsp_extractOrder-WORKING\procOrder.pl
 Printed at 19:48 on 06 Feb 1999

```

1 # procorder.pl (c) 1998,1999 E-DiaLog, Inc. All Rights Reserved.
2 #
3 # PURPOSE DISCUSSION
4 #
5 # program accepts a raw email stream from MS Exchange Server via export to ADO-compliant datastore
6 # then uses email body preprocessing and extraction rules - tailored per campaign in ./cf/ - to construct order reports.
7 #
8 # cf files are campaign optimized:
9 #
10 #   mailingPbLayout.cf          -> field mappings for a campaign's master mailing table
11 #   messageBodyDataLayout.cf    -> DSNs, field mappings, critical rule designations, order-designation bounds, synonymn table
12 #   messageBodyIndicateAddressChange.cf -> rulesets for indicating ADDRESS CHANGES
13 #   messageBodyIndicateBuyAll.cf -> rulesets for indicating BUY ALL state
14 #   messageBodyPreProcRules.cf  -> body text preprocessor configuration and rulesets
15 #
16 # program produces diagnostic and order-entry bound output files and logs outcomes of all ruleset evaluations leading to
17 # preparation of those output files, which include:
18 #
19 #   -> summary order report file (designed for data entry, faxing, emailing to fulfillment)
20 #   -> (companion) summary exception report (same format as s. order report with annotations to indicate rule failures for man
21 #   ual review)
22 #   OUT_<...>
23 #   EXC_DETAIL_<...>
24 #   EXC_BODY_<...>
25 #
26 # output files are (default) tab-delimited text
27 #
28 # PROGRAM REQUIRES
29 #
30 #   Perl 5.003 interpreter later compiled for WIN32
31 #   MS ADO components 1.5 (2.0+ recommended)
32 #
33 # RELEASE HISTORY
34 #
35 # 981020AE: 0.5      first usage: to extract carol10 order reports
36 # 981204AE: 0.9      upgrade to handle carol12, multiple item selections
37 # 981208AE: 1.1      improve flagging
38 # 981211AE: 1.5      adapt to extract IDEAS@WORK add/changes
39 # 981222AE: 2.3.1    fix bugs in EXC_BODY reporting / enhance it
40 # 981222AE: 2.4      add cFPATH support
41 # 981222AE: 2.4.1    cFpath logging; fix synonymn (SUBJECT TOKEN) lookup <trailing \s*>
42 # 981223AE: 2.5      reorder reporting fields
43 # 981224AE: 2.5.1    update SHIPPING_ADDRESS fields to reflect presence of T, B & S diagnostic fields
44 # 981225AE: 2.5.2    code review/dox; update cf to require MAILINGID
45 # 981226AE: 2.6      standardize on LOG ID for use in address change
46 # 990106AE: 2.6.5    add support for separate billing, shipping addresses
47 # 990129AE: 2.7      add record-recovery via ADO -> mailing database lookups
48 # 990201ae: 3.0      record-update ADO -> mailing database
49 # 990202ae: 3.0.1    fix leading digits bug in getDateinStamp fn
50 # 990202ae: 3.0.2    fix MOB query bugs; introduce detection of SELECT && UPDATE SUCCESS; verify decision-tree based UPDATES
51 # 990203ae: 3.1      fix address3 bug; provide regression compatibility for /ADDCHANGE
52 # 990205ae: 3.1.1    throw .notok as a HARD_EXC
53 # 990206ae: 3.1.2
54 #
55 # ##### INCL #####
56 # use win32::OLE;
57 #
58 # ##### ENV #####
59 #
60 # $requiredARG = 4;
61 # $Grevision   = '3.1.2';
62 #
63 # if($ARGV[0] < $requiredARG) { die "($Grevision) usage: procOrder.pl [userName] [reportName] [cfPATH <campaignMnemonic>] [ <buyToken> | /multi | /add
64 #   Change ]\n"; }
65 #
66 # $user
  
```

APPENDIX D

F:\Development\PRJ\PRJ_hbsp_extractOrder-WORKING\proOrder.pl
 Printed at 19:48 on 06 Feb 1999

```

67 $greportFile
68 $cPath
69 $subi_buyToken
70 $excreportFile
71 $excbodysubFile
72 $outFile
73 $goutFile
74
75 if($ARGV[4] == 1) { $gdebug = 1; } else { $gdebug = 0; }
76 if($ARGV[4] == 2) { $gdebug_FTN = 1; } else { $gdebug_FTN = 0; }
77
78 ##### DEFN #####
79
80 ## MISC GLOBALS ##
81
82 $platform = 'WIN';
83 $eventtype = 'INFO';
84 $basepath = '.';
85 $fielddelimiter = '\t';
86 $space = ' ';
87 $newline = '\n';
88 $comma = ',';
89 $pipe = '|';
90 $seventrequiresFUP = 'MULTI';
91 $multiItemOrderSwitch = '/ADDCCHANGE';
92 $caddChangeswitch = '/ADDCCHANGE';
93
94 ##### OUTPUT FILE LAYOUTS #####
95
96 ## DO NOT change alpha ordering of lowercase field names (e..t.)
97 ## This is vital in the BODY field layout defn file since the rules reference the letter prefixes
98
99 ## reportFile (note: EXC_reportFileName also written from this template)
100
101 %greportFileRecord = (
102
103   a_LID
104   ab_LINE
105   b_RID
106   c_THIS_SUBJECT
107   d_THIS_BODY_BOTTOM
108   e_THIS_BODY_TOP
109   fa_FUP_REQUIRED
110   fb_LOCATION_BUY_TOKEN
111   j_REVIEW_ACTION_TAKEN
112   za_SELECTED_ITEMS
113   zd_PRIORITY
114   zf_CUSTNO
115   zi_LISTID
116   zm_FROM_ADDRESS
117   zp_SPECIAL_INSTRUCTIONS
118   zr_PHONE
119   zx_BILLING_ADDRESS
120   zy_SHIPPING_ADDRESS
121   zz_MDB_BILLING_ADDRESS
122
123 );
124
125 ## outFile
126
127 %goutFileRecord = (
128
129   a_LID
130   aa_RID
131   ac_THIS_SUBJECT
132   ad_THIS_BODY_BOTTOM
133   b_LISTID
134   bb_CUSTNO
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

F:\Development\PRJ\PRJ_hbsp_extract\order-WORKING\procOrder.pl
 Printed at 19:48 on 06 Feb 1999

```

136   C_PRIORITY
137   e_first
138   f_bmiddle
139   q_blast
140   h_btitle
141   i_bcompany
142   j_bdepartment
143   k_baddress1
144   l_baddress2
145   m_baddress3
146   n_bcity
147   o_bstate
148   p_bpostal
149   q_bcountry
150   r_bphone
151   s_bfax
152   t_bemail
153
154   ue_first
155   uf_smiddle
156   ug_slash
157   uh_stitle
158   uj_scompany
159   uj_sdepartment
160   uk_saddress1
161   ul_saddress2
162   um_saddress3
163   un_scity
164   uo_sstate
165   up_spostal
166   uq_scountry
167   ur_sphone
168   us_sfax
169   ut_semail
170
171   ux_from_address
172   uy_location_buy_token
173   v_subject
174   z_fup_required
175   zz_review_type
176
177   );
178
179   ## exceptionsFile - BODY only
180   %exceptionsBODYFileRecord = (
181
182   =====>
183   a_LID
184   a_Review_Type
185   ab_Except_Flags
186   b_RID
187   u_from_address
188   v_subject
189   =====>
190   w_BODY_BELOW
191
192   );
193   ## activityLogFile
194   %activityLogRecord = (
195
196   a_LID
197   b_user
198   c_dateStamp
199   e_RID
200   f_clientName
201   g_eventType
202

```

```

=> '_e_$thismailingID',
=> '_e_$results\ue_first\ue_first',
=> '_e_$results\uf_smiddle\uf_smiddle',
=> '_e_$results\ug_slash\ug_slash',
=> '_e_$results\uh_stitle\uh_stitle',
=> '_e_$results\uj_scompany\uj_scompany',
=> '_e_$results\uj_sdepartment\uj_sdepartment',
=> '_e_$results\uk_saddress1\uk_saddress1',
=> '_e_$results\ul_saddress2\ul_saddress2',
=> '_e_$results\um_saddress3\um_saddress3',
=> '_e_$results\un_scity\un_scity',
=> '_e_$results\uo_sstate\uo_sstate',
=> '_e_$results\up_spostal\up_spostal',
=> '_e_$results\uq_scountry\uq_scountry',
=> '_e_$results\ur_sphone\ur_sphone',
=> '_e_$results\us_sfax\us_sfax',
=> '_e_$results\ut_semail\ut_semail',
=>
=> '_e_$results\ue_first\ue_first',
=> '_e_$results\uf_smiddle\uf_smiddle',
=> '_e_$results\ug_slash\ug_slash',
=> '_e_$results\uh_stitle\uh_stitle',
=> '_e_$results\uj_scompany\uj_scompany',
=> '_e_$results\uj_sdepartment\uj_sdepartment',
=> '_e_$results\uk_saddress1\uk_saddress1',
=> '_e_$results\ul_saddress2\ul_saddress2',
=> '_e_$results\um_saddress3\um_saddress3',
=> '_e_$results\un_scity\un_scity',
=> '_e_$results\uo_sstate\uo_sstate',
=> '_e_$results\up_spostal\up_spostal',
=> '_e_$results\uq_scountry\uq_scountry',
=> '_e_$results\ur_sphone\ur_sphone',
=> '_e_$results\us_sfax\us_sfax',
=> '_e_$results\ut_semail\ut_semail',
=>
=> '_e_$thisFromAddress',
=> '_e_$thisBuyStatus',
=> '_e_$thisSubject',
=> '_e_$eventRequiresFUP',
=> '_e_$EXCEPTION_TYPE',
=>
=> '_e_('$thisLogKey','<<')',
=> '_e_$EXCEPTION_TYPE',
=> '_e_$eventRequiresFUP',
=> '_e_('<<','$thisRecordID','>>')',
=> '_e_('<<','$thisFromAddress','>>')',
=> '_e_('$thisSubject','>>')',
=>
=> '_e_$thisFullNumberedBody',
=>
=> # full body
=>
=> '_e_$thisLogKey',
=> '_e_$user.'/'.<cfPath',
=> '_e_$dateStamp('\WIN',<GDEBUG_FTN)',
=> '_e_$thisRecordID',
=> '_e_$clientName',
=> '_e_$eventType',
=>

```

F:\Development\PRJ\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999

```

203 h_eventDesc => '_e_eventDesc',
204 z_fup_required => '_e_fupRequiredFUP',
205 );
206
207 ##### DATA #####
208 ##### # identify the fieldwise position of fields in the outfile for canonicalization (use EXC file as model)
209 #
210 #getkeyPos_Outfile(\%goutFileRecord);
211 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
212 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
213 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
214 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
215 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
216 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
217 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
218 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
219 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
220 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
221 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
222 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
223 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
224 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
225 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
226 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
227 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
228 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
229 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
230 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
231 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
232 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
233 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
234 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
235 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
236 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
237 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
238 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
239 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
240 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
241 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
242 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
243 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
244 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
245 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
246 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
247 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
248 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
249 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
250 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
251 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
252 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
253 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
254 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
255 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
256 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
257 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
258 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
259 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
260 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
261 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
262 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
263 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
264 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
265 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
266 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
267 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
268 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
269 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
270 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
271 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";

```

F:\Development\PRJ\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999

```

272 }
273
274 ## open I/O
275 if (open(FH_log, ">>$activityLogFileName")) {
276     {SeventType = 'ABORT'; $eventDesc = "MAIN::Cannot open GactivityLogFileName: \[$fileName\]"; &logE
277     vent(*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
278 if (open(FH_out, ">$outFile")) {
279     {SeventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GoutFile: \[$outFile\]"; &logEvent(*
280     $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
281 if (open(FH_report, ">$reportFile")) {
282     {SeventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GreportFile: \[$reportFile\]"; &logEve
283     nt(*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
284 if (open(FH_excReport, ">$excReportFile")) {
285     {SeventType = 'FAIL'; $eventDesc = "MAIN::Cannot open excReportFile: \[$excReportFile\]";
286     &logEvent(*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
287 if (open(FH_excBodyReport, ">$excBodyReportFile")) {
288     {SeventType = 'FAIL'; $eventDesc = "MAIN::Cannot open excBODYReportFile: \[$excBODYReportFile\]";
289     &logEvent(*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
290 if (open(FH_exc, ">$exceptionsFile")) {
291     {SeventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GexceptionsFile: \[$exceptionsFile\]"; &log
292     Event(*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
293 }
294 ## load $results fieldName keys from layout cf file (now in %LAYOUT)
295 undef;
296 @fieldKeys = &getFieldKeys(%LAYOUT);
297 #print "-debug--\n"; foreach(@fieldKeys) { print "***.$_.| \n"; }
298
299 $numberRequiredFields = scalar(@fieldKeys);
300
301 if(!defined($LAYOUT{'orderProcessingDSN'})) { $LAYOUT{'orderProcessingDSN'} = $GDSN; }
302
303 print "*** Loading daily order data source: $LAYOUT{'orderProcessingDSN'}...\n";
304 @DB = &read_DSNTOXOH($LAYOUT{'orderProcessingDSN'}, $LGDEBUG_FTN);
305
306 $totalRecords = $#DB +1;
307 $clientName = $LAYOUT{'thisClientName'};
308
309 ## startup OK event
310
311 $eventType = 'X_INFO'; $eventDesc = "procorder R.$revision START OK - USER: $Guser/$GcfPath CLIENT: $clientName opDSN ($totalRecords) $LAYOUT{'orderP
312 rocessingDSN'} mdbDSN: $LAYOUT{'orderProcessingDSN'}->$LAYOUT{'thisMailingTable'} ($totalRecords recs) BODYdefn: $bodyDataLayoutFileName"; &logEvent(*
313 FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
314
315 print "\n\n** Status...\n\n";
316 print "USER:\t\t$Guser\n";
317 print "CLIENT:\t\t$clientName\n";
318 print "CFPATH:\t\t$GcfPath\n";
319 print "TOTAL RECORDS \t\t-> $LAYOUT{'orderProcessingDSN'}"; $totalRecords\n";
320 unless($ADD_CHANGE_ON) { print "MAILING TABLE SOURCE \t-> $LAYOUT{'thisMailingTable'}\n\n"; }
321 print "\n(Note: if this information is not correct, hit <CTRL> - C to abort and correct <you have 5 seconds>)\n\n"; sleep 5;
322
323 print "*** Begin processing...\n\n";
324
325 ## flag duplicate e-mail addresses in order stream
326 %dupMap = &checkDupsReturnMap(@DB, $GDEBUG);
327
328 #####
329 main loop ####
330 #####
331
332 $excpcCount = 0;
333 $warnCount = 0;
334 $outFileCount = 0;
335 $exceptionsFileCount = 0;
336 $excReportFileCount = 0;
337 $reportFileCount = 0;
338 $FUPCount = 0;
339 $reportLineCount = 0;
340
341 for $mainCount (0 .. $#DB) {
342

```

F:\Development\PRJ\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
 Printed at 16:48 on 06 Feb 1999

```

333 $thisRecordID
334 $eventRequiresUP
335 $RECORD_VALID
336 $thisBuyStatus
337 $thisShippingAddressBlock
338 $thisShippingAddressBlock
339 $thisShippingAddressBlock
340 $thisFullNumberedBody
341
342 ## get relevant fields, normalized, this record
343
344 $thisFromAddress
345 $thisFromAddress
346 $thisFromName
347 $thisFromName
348 $thisSubject
349 $thisSubject
350 $thisSubject
351 $thisSubject
352 $thisBody
353 $thisBody
354 $thisBody
355
356 ## split BODY into discrete lines
357
358 @BODYlines
359 @topLines
360 @bottomLines
361
362 push(@BODYlines, $!) while ($thisBody =~ s/^(\\[012\\015]*)(\\012\\015?|\\015\\012?)/)/);
363
364 ## skip lines that are the quoted BODY unless they're the order form -- example delimiters:
365 ## $BODYstartToken = 'HARVARD BUSINESS SCHOOL PUBLISHING';
366 ## $BODYendToken = 'FIRST NAME\\.';
367
368 $BODYstartToken = $LAYOUT{'startDelimiter'};
369 $BODYendToken = $LAYOUT{'endDelimiter'};
370 $BODYstartToken =~ s/[\\t\\n\\r]/g;
371 $BODYendToken =~ s/[\\t\\n\\r]/g;
372 $startToken_ON = 0;
373 $endToken_ON = 0;
374 $thisBodyTop = undef;
375 $thisBodyBottom = undef;
376 $thisBodyExcluded = undef;
377
378 for($count = 0; $count <= $#BODYlines; $count++) {
379   $line
380   $thisFullNumberedBody .= ("{$count."} \\t.".$line."\\n");
381   unless($startToken_ON || $line =~ /\\>+/) {unless ($line =~ /\\>+/) {$thisBodyTop .= ($line.$gpipe)}; push(@topLines, ("{$count."} |'.$line.'|')}
382   ); }
383   if(($line =~ /$BODYstartToken/) .. ($line =~ /$BODYendToken/)) {
384     if(!($startToken_ON && ($line =~ /$BODYstartToken/))) {$startToken_ON = 1;}
385     if(!($endToken_ON && ($line =~ /$BODYendToken/))) {$endToken_ON = 1; $thisBody .= ($line.$gpipe)}; unless($line =~ /\\>+/)
386     || $line =~ /\\>+/ || $line =~ /\\[\\]/+/) {$thisBodyBottom .= ($line.$gpipe)} }
387     next;
388   }
389   ## ignore bottom lines if part of the original -> indicators: ">" || "[" || "]"
390   if($startToken_ON && $endToken_ON) { unless($line =~ /\\>+/ || $line =~ /\\[\\]/+/) {$thisBodyBottom .= ($line.$gpipe)}; push(@bottomLines, ("{$count."} |'.$line.'|')} }
391   $thisBody .= ($line.$gpipe);
392 }
393
394
395
396
397
398

```

```

399 $thisbodyTop =~ s/\\t\\r/\\_\\_g;
400 $thisbodyBottom =~ s/\\t\\r/\\_\\_g;
401
402 if ($GDEBUG) { print "\n\n\n** DEBUG___THISBODY=".$thisbody."\\n\\n\\n"; }
403
404 ## log/print warning if cannot find FOOTER ID BLOCK: CUSTNO|LISTID|MAILINGID|DIALOGID || if bad match yields the same for CUSTNO|LISTID
405
406 $thisCustNO = -1;
407 $thisListID = -1;
408 $thisMailID = -1;
409 $thisDialogID = -1;
410 $thisListID_EXCEPT_ON = 0;
411 $thisMailID_EXCEPT_ON = 0;
412 $thisDialogID_EXCEPT_ON = 0;
413
414 if ($thisbody =~ /\[([A-Z]*?)\]([A-Z]*?)\|([A-Z]*?)\|?([A-Z]*?)\|\\|\\|/) {
415     $LISTCID_EXCEPT_ON = 1;
416
417     $eventtype = 'EX_ERR'; $expcpcount++; $eventdesc = " \<" . $thisrecordID . "\> UNMATCHED FOOTER ID BLOCK: CUSTNO|LISTID|MAILINGID|DIALOGID";
418     &logEvent(\*FH_log, $eventtype, $eventdesc, $guser, $LGDEBUG_FTN);
419     $eventtype = 'INFO'; $expcpcount++; $eventdesc = " \<" . $thisrecordID . "\> >>> ATTEMPTING SUBJECT LINE MAILING ID RECOVERY >>>"; &logEvent(
420     \*FH_log, $eventtype, $eventdesc, $guser, $LGDEBUG_FTN);
421
422     ## test to see if there is a cf file synonym (**,++, etc) for the MAILING ID which can be recovered in the absence of the footer token
423     if ($thismailingID == -1) { $thismailingID = &subjectSynonymLookup($thissubject, %LAYOUT); }
424     if ($thismailingID == -1) {
425         RECORD_VALID=0;
426         $MAILINGID_EXCEPT_ON = 1;
427         $eventtype = 'EX_ERR'; $expcpcount++; $eventdesc = " \<" . $thisrecordID . "\> MAILING ID RECOVERY WAS NOT SUCCESSFUL"; &logEvent(\*F
428         H_log, $eventtype, $eventdesc, $guser, $LGDEBUG_FTN);
429     }
430
431     if ($thismailingID != -1) {
432         $eventtype = 'INFO'; $expcpcount++; $eventdesc = " \<" . $thisrecordID . "\> <<< MAILING ID RECOVERY SUCCESSFUL USING: $thissubjectma
433         ilingIDToken -> $LAYOUT{$thissubjectmailingIDToken}"; &logEvent(\*FH_log, $eventtype, $eventdesc, $guser, $LGDEBUG_FTN);
434     }
435
436     else {
437         if ($GDEBUG) { print "***DEBUG_ element1= $1 ___ element2= $2 ___ element3= $3 ___ element4= $4\\n"; sleep 1; }
438
439         $element1 = $1; $element2 = $2; $element3 = $3; $element4 = $4;
440
441         $thisCustNO = ($element1 =~ /\S+?/) ? $element1 : -1;
442         $thisListID = ($element2 =~ /\S+?/) ? $element2 : -1;
443         $thisMailID = ($element3 =~ /\S+?/) ? $element3 : -1;
444         $thisDialogID = ($element4 =~ /\S+?/) ? $element4 : -1;
445
446         if ($thisCustNO == -1 && $thisListID == -1) {
447             $LISTCID_EXCEPT_ON = 1;
448             $MAILINGID_EXCEPT_ON = 1;
449             $eventtype = 'EX_ERR'; $expcpcount++; $eventdesc = " \<" . $thisrecordID . "\> BAD MATCH: CUSTNO, LISTID"; &logEvent(\*FH_log, $even
450             ttype, $eventdesc, $guser, $LGDEBUG_FTN);
451         }
452
453         ## log/print INFO if cannot find mailingID
454         if ($thismailingID == -1) { $eventtype = 'INFO'; $eventdesc = " \<" . $thisrecordID . "\> NO MAILING ID MATCH"; &logEvent(\*FH_log, $eventtyp
455         e, $eventdesc, $guser, $LGDEBUG_FTN); }
456
457         ## log/print INFO if cannot find dialogID
458     }
459
460 }
461

```


Page 8

F:\Development\PRAPRA_hpsp_extractOrder-WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999

```

462         if($thisDialogID == -1) { $eventType = 'INFO'; $eventDesc = " \<"$thisRecordID."> NO DIALOG ID MATCH"; &logEvent("\*FH_log, $eventType,
463         $eventDesc, $guser, $LGDEBUG_FTN);}
464     }
465     ## check for /multi || quoted BUY TOKEN (eg: "YES")
466     $buyTokenPosition = 'none';
467     $thisItemsSelectedStack = undef;
468     @thisItemsSelected = undef;
469     unless($ADD_CHANGE_ON) {
470         ## check for buy tokens, eg: YES in SUBJECT, TOP, BOTTOM || letter items: ABC || A,B,C || A-C
471         $buyTokenPosition = &detectBuyTokens($MULTI_BUY_ON,$gsubj_buyToken,$thissubject,$thismailingID,$thisbodyTop,$thisbodyBottom,\%LAYO
472         UT,\@topLines,\@bottomLines,$GDEBUG);
473         ## check for no-match case and except
474         if($buyTokenPosition eq 'none') { $eventRequiresFUP .= '.noTok'; $FUPcount++; $eventType = 'ERR_FUP'; $warnCount++; $eventDesc =
475         " \<"$thisRecordID."> SUBJECT, TOP, BOTTOM -> NO BUY TOKEN MATCH ($gsubj_buyToken); &logEvent("\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN)
476     };
477     ## if match and MULTI_BUY_ON, prepare order items listing
478     if($buyTokenPosition ne 'none' && $MULTI_BUY_ON) {
479         $thisItemsSelectedStack = &extractDedupItemsSelectedStack(@thisItemsSelected);
480     }
481     #####
482     ## INNER LOOP ##
483     #####
484     %results = {};
485     $countFields = 0;
486     $nullCriticalFields = 0;
487     $ADDRESS_ADEQUATE = 1;
488     ## process body fields by key; keys == standard field names; values == this BODY's field names mapped thereon
489     foreach (@fieldKeys) {
490         $key = $_;
491         ## print "--DEBUG-- key = |$key| \n";
492         $thisLayoutFieldName = uc($LAYOUT{$key});
493         $thisLayoutFieldName =~ s/\s+//;
494         #####
495         ##### MAIN MATCHING BLOCK #####
496         #####
497         $MATCH_DATA_ON = 0;
498         $RULE = -1;
499         $expectedMatch = $thisLayoutFieldName.'\.:.*?'\{([\[]+?)\}';
500         $noBracketsMatch = $thisLayoutFieldName.'\.:.*?'\{([\[]+?)\}';
501         CASE: {
502             TRY_EXPECTED_MATCH: if($thisBody =~ /$expectedMatch/)
503             TRY_NO_BRACKETS_MATCH: if($thisBody =~ /$noBracketsMatch/)
504             CASE; }
505     }
506     { $MATCH_DATA_ON = 1; $matchElement = $1; $RULE = 1; last
507     { $MATCH_DATA_ON = 1; $matchElement = $1; $RULE = 2; last

```

Page 9

T:\Development\PR\PR11\hisp-extract\order-WORKING\procOrder.pl
 Printed at 19:48 on 06 Feb 1999

```

525     }
526
527     ## A MATCH WAS FOUND, THIS KEY ... push into results set
528     if ($MATCH_DATA_ON) {
529         $countFields++;
530         $matchElement =~ s/[^\w\s\-\.\_\#\!\@\]/$Gspace/g;
531         $results{$key} = &normalizeSpacing($matchElement);
532
533         ## print "--DEBUG-- result = |results{$key}| \n";
534
535         ## build shipping and billing address blocks
536         CASE: {
537             if ($key =~ /u[a-z]_/) {
538                 ## construct SHIPPING ADDRESSBLOCK for report
539                 if ($key =~ /(hijk|mq)\_/) { $thisShippingAddressBlock .= (length($results{$key}) > 0)?($results{$key}).$Gpipe:'';
540                 if ($key =~ /(efgnop)\_/) { $thisShippingAddressBlock .= (length($results{$key}) > 0)?($results{$key}).$Gspace:'';
541             }
542             last CASE;
543         }
544         if ($key =~ /[nu]_/) {
545             ## construct BILLING ADDRESSBLOCK for report
546             if ($key =~ /(hijk|mq)\_/) { $thisBillingAddressBlock .= (length($results{$key}) > 0)?($results{$key}).$Gpipe:'';
547             if ($key =~ /(efgnop)\_/) { $thisBillingAddressBlock .= (length($results{$key}) > 0)?($results{$key}).$Gspace:'';
548         }
549         last CASE;
550     }
551
552     ## check for null critical fields: any absence here will throw exception, unless CUSTNO
553     if ($key =~ /criticalFields/) { \n";sleep 1;
554     if ($key =~ /criticalFields/) { \n";sleep 1;
555     if ($key =~ /criticalFields/) { \n";sleep 1;
556     if ($key =~ /criticalFields/) { \n";sleep 1;
557     if ($key =~ /criticalFields/) { \n";sleep 1;
558     if ($key =~ /criticalFields/) { \n";sleep 1;
559     if ($key =~ /criticalFields/) { \n";sleep 1;
560     if ($key =~ /criticalFields/) { \n";sleep 1;
561     if ($key =~ /criticalFields/) { \n";sleep 1;
562     if ($key =~ /criticalFields/) { \n";sleep 1;
563     if ($key =~ /criticalFields/) { \n";sleep 1;
564     if ($key =~ /criticalFields/) { \n";sleep 1;
565     if ($key =~ /criticalFields/) { \n";sleep 1;
566     if ($key =~ /criticalFields/) { \n";sleep 1;
567     if ($key =~ /criticalFields/) { \n";sleep 1;
568     if ($key =~ /criticalFields/) { \n";sleep 1;
569     if ($key =~ /criticalFields/) { \n";sleep 1;
570     if ($key =~ /criticalFields/) { \n";sleep 1;
571     if ($key =~ /criticalFields/) { \n";sleep 1;
572     if ($key =~ /criticalFields/) { \n";sleep 1;
573     if ($key =~ /criticalFields/) { \n";sleep 1;
574     if ($key =~ /criticalFields/) { \n";sleep 1;
575     if ($key =~ /criticalFields/) { \n";sleep 1;
576     if ($key =~ /criticalFields/) { \n";sleep 1;
577     if ($key =~ /criticalFields/) { \n";sleep 1;
578     if ($key =~ /criticalFields/) { \n";sleep 1;
579     if ($key =~ /criticalFields/) { \n";sleep 1;
580     if ($key =~ /criticalFields/) { \n";sleep 1;
581     if ($key =~ /criticalFields/) { \n";sleep 1;
582     if ($key =~ /criticalFields/) { \n";sleep 1;
583     if ($key =~ /criticalFields/) { \n";sleep 1;
584     if ($key =~ /criticalFields/) { \n";sleep 1;
585     if ($key =~ /criticalFields/) { \n";sleep 1;
586     if ($key =~ /criticalFields/) { \n";sleep 1;

```

F:\Development\PRJ\PRJ_hbsp_extract\order-WORKING\propOrder.pl
 printed at 19:48 on 06 Feb 1999

```

587   if ($key =~ /$LAYOUT{'criticalFields'}\\) {
588     $RECORD_VALID=0; $ADDRESS_ADEQUATE = 0;
589     $eventRequiresFUP .= ((($eventRequiresFUP =~ /badad/)?'':'.badad'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $event
590     tDesc = " \<"$thisRecordID."> UNPARSABLE STANDARD FIELD_NAME=FIELD_VALUE PAIR IS CRITICAL: \{thisLayoutFieldName\}"; &logEvent(\*FH_log, $eventType,
    $eventDesc, $guser, $LGDEBUG_FTN);
591   }
592 }
593
594 #####
595 ##### END INNER LOOP ###
596 #####
597 ##### SUMMARY RULES ###
598 #####
599 #####
600 #####
601 ##### SUMMARY RULES ###
602 #####
603
604 ## throw EX_LISTNO/CUSTNO error exception if address is not OK && iPhone, depending on CID_LID_REQUIRED from cf file
605
606   if ($LAYOUT{'CID_LID_REQUIRED'}) {
607     if ($LISTCID_EXCEPT_ON) {
608       $RECORD_VALID = 0;
609       $eventRequiresFUP .= ((($eventRequiresFUP =~ /chkCLID/)?'':'.chkCLID'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $eventDes
610       c = " \<"$thisRecordID."> EITHER CID || LID ABSENT WHEN CF CRITICAL"; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
611     }
612   }
613
614   if (! $LAYOUT{'CID_LID_REQUIRED'}) {
615     if ($LISTCID_EXCEPT_ON && !($ADDRESS_ADEQUATE && $results{'r\phone'} =~ /\s+?/)) { $RECORD_VALID = 0; }
616   }
617
618   if ($LAYOUT{'MAILINGID_REQUIRED'}) {
619     if ($MAILINGID_EXCEPT_ON) {
620       $RECORD_VALID = 0;
621       $eventRequiresFUP .= ((($eventRequiresFUP =~ /chkMID/)?'':'.chkMID'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $eventDesc
622       = " \<"$thisRecordID."> MAILING ID ABSENT WHEN CF CRITICAL"; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
623     }
624   }
625
626   if ($LAYOUT{'MAILINGID_REQUIRED'}) {
627     if ($MAILINGID_EXCEPT_ON) {
628       $RECORD_VALID = 0;
629       $eventRequiresFUP .= ((($eventRequiresFUP =~ /chkMID/)?'':'.chkMID'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $eventDesc
630       = " \<"$thisRecordID."> MAILING ID ABSENT WHEN CF CRITICAL"; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
631     }
632   }
633
634   if (($countFields < $numberRequiredFields) && $thisCustNo < 1) {
635     $eventType = 'INFO'; $eventDesc = " \<"$thisRecordID."> (OVERALL) NOT ALL STANDARD FIELDS FOUND: $countFields ($numberRequiredFields)"
636     ; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
637   }
638
639   # throw INFO if less than number required
640
641   # tag duplicates for FUP
642
643   if ($dupsmap{$mainCount} =~ /\#\#\#+\#\#/) { $eventRequiresFUP .= ((($eventRequiresFUP =~ /dupad/)?'':'.dupad'); $FUPcount++; }
644
645   # catch thisBodyTop and thisBodyBottom blank yet address still extracted adequately (eg happens when cut & paste responses between the body delimiters)
646
647   if ($RECORD_VALID && $thisBodyTop =~ /\s+?/ && $thisBodyBottom =~ /\s+?/) {
648
649
650

```

Page 11

FA Development\PR\PRR\prp\prp\extra\order\WORKING\prp\order.n
Printed at 19:48 on 06 Feb 1999

```

651 $RECORD_VALID = 0;
652 $eventtype = 'EX_ERR'; $eventrequiresfup .= '.chkbody'; $fupcount++; $eventdesc = " \<" $thisrecordid. "\> CUSTOMER RESPONSE (Top,
Bottom) NOT FOUND WHEN EXPECTED"; &logevent("\*FH_log, $eventtype, $eventdesc, $guser, $LGDEBUG_FTN);
653
654 }
655 #####
656 ##### RECORD INVALID: Attempt recovery from mailing database #####
657 #####
658 #####
659 $RECORD_RECOVERED = 0;
660 $RECORD_VALID && !$ADD_CHANGE_ON) {
661   if (!$thisrecordid. "\> >>> BAD RECORD: ATTEMPTING RECOVERY-LOOKUP USING \"$thisfromaddress\", $LAYOUT{'thisma
ilingdsn'} -> $LAYOUT{'thismailingtable'}; &logevent("\*FH_log, $eventtype, $eventdesc, $guser, $LGDEBUG_FTN);
662   $eventtype = 'INFO'; $eventdesc = " \<" $thisrecordid. "\>
663   $eventtype = 'EX_ERR'; $eventrequiresfup .= ((($eventrequiresfup =~ /badmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventdesc
ilingdsn') -> $LAYOUT{'thismailingtable'}; &logevent("\*FH_log, $eventtype, $eventdesc, $guser, $LGDEBUG_F
664   $recoveredresults = undef;
665   $recoveredresults = &queryselect_db("\%LAYOUT, 'EMAIL', $thisfromaddress, $LGDEBUG_FTN);
666   $recoveredresults = &queryselect_db("\%LAYOUT, 'EMAIL', $thisfromaddress, $LGDEBUG_FTN);
667   foreach(keys($recoveredresults)) { print "debug== key = $_ ____ value = $recoveredresults{$_} \n"; }
668   CASE: {
669     if($recoveredresults{'_exit_'} < 1) {
670       $RECORD_VALID = 0;
671       $eventtype = 'EX_ERR'; $eventrequiresfup .= ((($eventrequiresfup =~ /badmdb/) ? '' : '.badmdb'); $fupcount++; $excpcount++; $eventdesc
672       $eventtype = 'EX_ERR'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
673       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
674       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
675       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
676       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
677       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
678       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
679       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
680       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
681       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
682       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
683       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
684       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
685       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
686       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
687       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
688       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
689       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
690       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
691       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
692       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
693       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
694       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
695       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
696       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
697       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
698       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
699       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
700       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
701       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
702       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
703       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
704       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
705       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
706       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
707       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
708       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
709       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
710       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde
711       $eventtype = 'WRN_FUP'; $eventrequiresfup .= ((($eventrequiresfup =~ /goodmdb/) ? '' : '.goodmdb'); $fupcount++; $excpcount++; $eventde

```

T:\Development\PAULPB\hbsp_extra_order-WORKING\procode.pl
Printed at 19:48 on 06 Feb 1999

```

712 }
713 ### add an empty flag where SHIPPING ADDRESS is returned blank
714
715 if($thisshippingaddressblock !~ /\s+$/) { $thisshippingaddressblock = '{ SAME }'; }
716 if($thismdbillingaddressblock !~ /\s+$/) { $thismdbillingaddressblock = '{ N/A }'; }
717
718 ### tag address block exceptions with diagnostic ques
719
720 if(! $RECORD_VALID) {
721     CASE: {
722         if($thisbillingaddressblock !~ /\s+$/ && ($thisbodytop =~ /\s+$/ || $thisbodybottom =~ /\s+$/))
723             { $thisbillingaddressblock = 'NO_'; }
724         if($thisshippingaddressblock !~ /\s+$/ && $thisbodytop !~ /\s+$/ && $thisbodybottom !~ /\s+$/))
725             { $thisshippingaddressblock = '{ BO'; }
726         if($thisbillingaddressblock !~ /\s+$/ && ! $ADDRESS_ADEQUATE)
727             { $thisbillingaddressblock = 'CRIT'; }
728         if($thisbillingaddressblock !~ /\s+$/ && ! $ADDRESS_ADEQUATE)
729             { $thisbillingaddressblock = 'UNKNOWN_ERR/INSPECT='; $thisbillingaddressblock; last CASE; }
730     }
731 }
732
733 $eventrequiresfup = ($eventrequiresfup !~ /\s+$/) ? 'inspect' : $eventrequiresfup;
734
735 #####
736 ## write valid record data to outfiles ##
737 #####
738 #####
739
740 $ED_ORDER_VALUE = 1;
741 $EXCEPTION_TYPE = 'ok';
742 $UPDATE_SUCCESSFUL = 0;
743 $THIS_KEY_FIELD = undef;
744 $THIS_KEY_FIELD_VALUE = undef;
745
746 if($RECORD_VALID) {
747     unless($ADD_CHANGE_ON) {
748         CASE: {
749             if(&queryUPDATE_DB(\%LAYOUT, $THIS_KEY_FIELD_VALUE = $thisfromaddress, $THIS_KEY_FIELD = "EMAIL", "ED_ORDER", $ED_ORDER_VALUE,
750                 $UPDATE_SUCCESSFUL=1; last CASE; });
751             if(&queryUPDATE_DB(\%LAYOUT, $THIS_KEY_FIELD = "CID", "ED_ORDER", $ED_ORDER_VALUE, "ED_MAI
752                 LACTION", -1, $RECORD_RECOVERED, $LGDEBUG_FTN)) { $UPDATE_SUCCESSFUL=1; last CASE; }
753             if(&queryUPDATE_DB(\%LAYOUT, $THIS_KEY_FIELD = $thislistid, "ED_ORDER", $ED_ORDER_VALUE, "ED_
754                 MAILACTION", -1, $RECORD_RECOVERED, $LGDEBUG_FTN)) { $UPDATE_SUCCESSFUL=1; last CASE; }
755         }
756     }
757 }
758
759 if(! $UPDATE_SUCCESSFUL) {
760     $RECORD_VALID = 0;
761     $eventtype = 'EX_ERR'; $eventrequiresfup .= (($eventrequiresfup =~ /badupmdb/?'':'.badupmdb') : $fupcount++ : $excpcount++ : $event
762         desc = " \<" . $thisrecordid . "\> (MANUAL UPDATE REQUIRED) MAILING TABLE UPDATE FAILED ON ALL KEYS - LAST TRY: \" $THIS_KEY_FIELD -> $THIS_KEY_FIELD_VALUE\"
763         "; &logEvent(\%FH_log, $eventtype, $eventdesc, $user, $LGDEBUG_FTN);
764     }
765 }
766
767 if(! $RECORD_VALID && ! $UPDATE_SUCCESSFUL) {
768     $eventtype = 'EX_ERR'; $eventrequiresfup .= (($eventrequiresfup =~ /noupmdb/?'':'.noupmdb') : $fupcount++ : $excpcount++ : $eventdesc = " \
769         <" . $thisrecordid . "\> INVALID RECORD: MAILING TABLE NOT UPDATED"; &logEvent(\%FH_log, $eventtype, $eventdesc, $user, $LGDEBUG_FTN);
770     }
771 }

```

F:\Development\PR\PRJ_hbsp_extract\order-WORKING\proccorder.pl
 Printed at 19:48 on 06 Feb 1999

```

772 }
773 }
774
775 ### Buy tokens not found and not ADDRESS_CHANGE; move to EXC report
776 unless($ADDRESS_CHANGE) { if($buyTokenPosition eq 'none') { $RECORD_VALID = 0; $ED_ORDER_VALUE = 2; } }
777
778 if($RECORD_VALID) {
779     $reportLineCount++;
780     ## update MAILING TABLE
781
782     ## write out files
783     &writerRecord(\*FH_out,$gfielddelimiter,($thiskey=&getNextDBkey_return($goutFileDBkeysFilename,$LGDEBUG_FTN)),\%goutFileRecord,'EXTENDED',
784     $LGDEBUG_FTN);
785     &writerRecord(\*FH_report,$gfielddelimiter,($thiskey=&getNextDBkey_return($greportFileDBkeysFilename,$LGDEBUG_FTN)),\%greportFileRecord,'NO
786     RMAL', $LGDEBUG_FTN);
787
788     ## include ALL BODYS in the EXC_BODY file for potential review
789     if($eventRequiresFUP =~ /\s+?/) { $EXCEPTION_TYPE = 'SOFT-EXC'; }
790
791     &writerRecord(\*FH_excBODYReport,$gfielddelimiter,$thisBODYlogkey=&getNextDBkey_return($exceptionsBODYDBkeysFilename,$LGDEBUG_FTN)
792     ,\%gexceptionsBODYFileRecord,'BODY', $LGDEBUG_FTN);
793
794     $outFileCount++;$reportFileCount++;
795 }
796
797 #####
798 ## write exception record data to outfiles ##
799 #####
800
801 if(!$RECORD_VALID) {
802     $EXCEPTION_TYPE = 'HARD-EXC';
803     &writerRecord(\*FH_exc,$gfielddelimiter,$thiskey=&getNextDBkey_return($exceptionsDBkeysFilename,$LGDEBUG_FTN),\%goutFileRecord,'EXTENDED',
804     $LGDEBUG_FTN);
805     &writerRecord(\*FH_excReport,$gfielddelimiter,$thiskey=&getNextDBkey_return($gexcReportDBkeysFilename,$LGDEBUG_FTN),\%greportFileRecord,'NO
806     ptionsBODYFileRecord,'BODY', $LGDEBUG_FTN);
807
808     $exceptionsFileCount++; $excReportFileCount++;
809
810     print "\n";
811 }
812
813 #####
814 ## end main loop ###
815 #####
816
817 ## print STDOUT exit state
818
819 $dateNow = &getDateStamp('WIN',$LGDEBUG_FTN);
820
821 print "\n** [proccorder version $Greversion] Done! $totalRecords records processed by $Guser/$GcPath for client: $GclientName **\n\n";
822 print "SUMMARY report for: ",$dateNow,"\n\n";
823 print "FILES WRITTEN\n";
824 print "\n%10d", $reportFileCount;
825 print "\n%10d", $recordsWrittenToReportFile,$GreportFile;
826 print "\n%10d", $excReportFileCount;
827 print "\n%10d", $recordsWrittenToExcReportFile,$GexcReportFile;
828

```

F:\Development\PRJ\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999

```

835 print " records written to OUTPUT FILE (all fields): $outFile";
836 printf "\n%10d", $exceptionsFilecount;
837 print " records written to EXCEPTIONS FILE (all fields): $exceptionsFile";
838 print "\n\nLOGGING\n";
839 print "all rule failure events for all time logged in detail to: $activityLogFile\n";
840 printf "\n%10d", $totalRecords;
841 print " BODYS for this session are keyed, categorized in: $excBODYreportFile, for review in an editor\n";
842
843 $eventType = 'x-INFO'; $eventDesc = "procOrder R $grevision EXIT OK - USER: $Guser/$GcfPath CLIENT: $GclientName opDSN ($totalRecords): $LAYOUT{'orderPr
ocessingDSN'} mdbDSN: $LAYOUT{'orderProcessingDSN'} -> $LAYOUT{'thisMailingTable'} ($totalRecords recs) BODYdefn: $GbodyDataLayoutFileName"; &logEvent(\
FH_log, $eventType, $eventDesc, $Guser, $LGDBUG_FIN);
844
845 close(FH_out); close(FH_exc); close(FH_log); close(FH_report); close(FH_excBODYreport);
846
847 exit;
848 #####
849 ##### SUBS #####
850 #####
851 #####
852 #####
853 # -----
854 #
855 #
856 #
857 sub alphaordered {
858     my(@singleElementArray) = @_;
859
860     my $alpha = 0;
861     my $i = 0;
862     my $base = $singleElementArray[0];
863
864     for($i = 1; $i <= $#singleElementArray; $i++) {
865         $ordBase = ord($base); $ordArr = ord($singleElementArray[$i]);
866         print "--debug-$i= base = |$base| ordBase= $ordBase ___ singleElementArray[$i] ordArr = $ordArr \n";
867         if($ordBase > $ordArr) { return $alpha; }
868         $base = $singleElementArray[$i];
869     }
870
871     $alpha = 1;
872     return $alpha;
873 }
874
875 sub canonicalize {
876     my($inRecord) = @_;
877     my($elementIndex, $fieldsIndex, $rec, $buf, $field) = undef;
878     my($@field) = undef;
879     my($space) = " ";
880     my($dashEscape) = $space.'_d_esc_'. $space;
881     my($inRecord) = uc($inRecord);
882     my @allFields = undef;
883     my @allFields = split(/$GfieldDelimiter/, $inRecord);
884     my $fieldsIndex = undef;
885     my $fieldsIndex=0; $fieldsIndex <= $#allFields; $fieldsIndex++ {

```

[illegible]

F:\Development\PRJ\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
 Printed at 16:48 on 06 Feb 1999

```

963 # -----
964 #
965 #
966 sub checkDupsReturnMap {
967   my($ptrAODB,$GDEBUG) = @_ ;
968   my($ptrAODB) = @$ptrAODB;
969   ## prepare map of duplicates on FromAddress
970   my %dupsMap = undef;
971   my %j (0 .. $#DB) {
972     $dupsMap{$j} = $DB[$j]['FromAddress']
973   }
974   my @mirrorDupsMap = undef;
975   my %key = undef;
976   my $value = undef;
977   my @arr = scalar(@arr);
978   foreach $key(keys %dupsMap) {
979     $value = $dupsMap{$key};
980     next if($value !~ /\s+?/);
981     @arr = grep /$value/, @mirrorDupsMap;
982     $number = scalar(@arr);
983     if($number > 1) {
984       $eventType = 'INFO'; $eventDesc = "DUPLICATE FromAddress ITEM DETECTED ON IMPORT & FLAGGED: ($number|$key|$value)"; &logEvent(\*FH_to
985       $eventType, $eventDesc, $user, $GDEBUG.FTN);
986       $dupsMap{$key} = ("###.$number.### '$dupsMap{$key}');
987       ## print $dupsMap{$key}."\\n";
988       ## print "*** key=".$key." *** value=".$dupsMap{$key}."___occurrences=".$number."\\n";
989     }
990   }
991   return %dupsMap
992 }
993
994 sub checkMultiBuy_Extract {
995   my($thisline,$thismailingID,$pos,$ptrLayout,$GDEBUG) = @_ ;
996   my($text) = 0;
997   my(%LAYOUT) = %$ptrLayout;
998   my(@rangeStack) = undef;
999   my(@singleStack) = undef;
1000   my($countRangeMatches) = 0;
1001   my($repeat) = 1;
1002   my($match) = undef;
1003   my($preProcmoveChar) = '%';
1004   my($preProcmoveChar) = '%';
1005   my($elementSeparator) = ',';

```

```

1031 ## PREPARE THE LINE FOR MATCHING
1032
1033 $thisline = uc($thisline);
1034
1035 ## RUN RULESET FROM RULES CF FILE
1036 foreach(@GpreProcRules) { eval; }
1037
1038 my $thisPreExtractionLine = $thisline;
1039 my @testSingleStack = undef;
1040 my $maxASCII = 0;
1041
1042 ## EXTRACTION LOOP
1043 while($repeat) {
1044     my $RULE = 0;
1045     my $match = undef;
1046     my $print "\n\n--debug-- maxASCII = $maxASCII -- thisline IS CURRENTLY |$thisline| \n\n";
1047     #MATCH_CASE: {
1048         if($thisline =~ /\{([A-Z])\}/) {
1049             if($thisline =~ /\{([A-Z])\}([AA-Z\^%\@\\-]{1})[AA-Z\^%\@\\-]{1}[AA-Z\^%\@\\-]*$/) {
1050                 $match = $1; $RULE = -1; last MATCH_CASE;
1051             }
1052             if($thisline =~ /\{([A-Z])\}([AA-Z\^%\@\\-]{1})[AA-Z\^%\@\\-]{1}[AA-Z\^%\@\\-]*$/) {
1053                 $match = $1; $RULE = -2; last MATCH_CASE;
1054             }
1055             if($thisline =~ /\{([A-Z])\}([AA-Z\^%\@\\-]{1})[AA-Z\^%\@\\-]{1}[AA-Z\^%\@\\-]*$/) {
1056                 $match = $1; $RULE = -3; last MATCH_CASE;
1057             }
1058         }
1059     }
1060     RANGE_CASE: {
1061         if($thisline =~ /\{([A-Z])\}([AA-Z\^%\@\\-]{1})[AA-Z\^%\@\\-]{1}[AA-Z\^%\@\\-]*$/) {
1062             if($thisline =~ /\{([A-Z])\}([AA-Z\^%\@\\-]{1})[AA-Z\^%\@\\-]{1}[AA-Z\^%\@\\-]*$/) {
1063                 $match = $1; $RULE = 1; last MATCH_CASE;
1064             }
1065             if($thisline =~ /\{([A-Z])\}([AA-Z\^%\@\\-]{1})[AA-Z\^%\@\\-]{1}[AA-Z\^%\@\\-]*$/) {
1066                 $match = $1; $RULE = 2; last MATCH_CASE;
1067             }
1068             if($thisline =~ /\{([A-Z])\}([AA-Z\^%\@\\-]{1})[AA-Z\^%\@\\-]{1}[AA-Z\^%\@\\-]*$/) {
1069                 $match = $1; $RULE = 3; last MATCH_CASE;
1070             }
1071             if($thisline =~ /\{([A-Z])\}([AA-Z\^%\@\\-]{1})[AA-Z\^%\@\\-]{1}[AA-Z\^%\@\\-]*$/) {
1072                 $match = $1; $RULE = 4; last MATCH_CASE;
1073             }
1074         }
1075     }
1076     $repeat = 0;
1077 }
1078
1079 $thisline =~ s/$match/$replacementEscChar/g;
1080
1081 $thisCheckedMailID = ($thismailingID=0) ? $thismailingID : -1;
1082
1083 ##print "\n- debug - ELEMENT CONSIDERED with break = $RULE |$match| \n";
1084 if($DEBUG) { print "--debug-- BEGIN RULES -- RANGE = |LAYOUT|$thisCheckedMailingID| from $thisCheckedMailingID \n"; }
1085
1086 ## BEGIN RULES TO SEE WHAT WE GOT ##
1087 $PUSH = 1;
1088 if($RULE > 0) {
1089     my($arr) = undef;
1090     my $tempMatch = $match;
1091     $tempMatch =~ s/\-//g;
1092     @arr = split(/,/,$tempMatch);
1093     if(!&lphaOrdered(@arr)) {
1094         ## print "-- debug -- REJECT - not alpha match |$match| in line |$thisPreExtractionLine| \n";
1095         $PUSH = 0;
1096     }
1097 }

```

```

1100 }
1101 #if($RULE > 0 && ord($arr[0]) <= $maxASCII) {
1102   # print "-- debug -- REJECT - range case [$match] out of alpha order with current stack!\n";
1103   # if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
1104   # $PUSH = 0;
1105   #
1106   #
1107   #
1108   #
1109   #
1110   #
1111   if(defined($match) && $match !~ /[\\$\\OUT{\\thisCheckedMailingID}]/) {
1112     ## print "--debug-- REJECT - not in RANGE = [\\$\\OUT{\\thisCheckedMailingID}] match [$match] in line [\\thisPreExtractionLine|\\n";
1113     if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
1114     $PUSH = 0;
1115     #
1116     #
1117     #
1118     #
1119     #
1120     #
1121     #
1122     #
1123     #
1124     #
1125     #
1126     #
1127     #
1128     #
1129     #
1130     #
1131     #
1132     #
1133     #
1134     #
1135     #
1136     #
1137     #
1138     #
1139     #
1140     #
1141     #
1142     #
1143     #
1144     #
1145     #
1146     #
1147     #
1148     #
1149     #
1150     #
1151     #
1152     #
1153     #
1154     #
1155     #
1156     #
1157     #
1158     #
1159     #
1160     #
1161     #
1162     #
1163     #
1164     #
1165     #
1166     #
1167     #
1168     #
1169     #
1170     #
1171     #
1172     #
1173     #
1174     #
1175     #
1176     #
1177     #
1178     #
1179     #
1180     #
1181     #
1182     #
1183     #
1184     #
1185     #
1186     #
1187     #
1188     #
1189     #
1190     #
1191     #
1192     #
1193     #
1194     #
1195     #
1196     #
1197     #
1198     #
1199     #
1200     #
1201     #
1202     #
1203     #
1204     #
1205     #
1206     #
1207     #
1208     #
1209     #
1210     #
1211     #
1212     #
1213     #
1214     #
1215     #
1216     #
1217     #
1218     #
1219     #
1220     #
1221     #
1222     #
1223     #
1224     #
1225     #
1226     #
1227     #
1228     #
1229     #
1230     #
1231     #
1232     #
1233     #
1234     #
1235     #
1236     #
1237     #
1238     #
1239     #
1240     #
1241     #
1242     #
1243     #
1244     #
1245     #
1246     #
1247     #
1248     #
1249     #
1250     #
1251     #
1252     #
1253     #
1254     #
1255     #
1256     #
1257     #
1258     #
1259     #
1260     #
1261     #
1262     #
1263     #
1264     #
1265     #
1266     #
1267     #
1268     #
1269     #
1270     #
1271     #
1272     #
1273     #
1274     #
1275     #
1276     #
1277     #
1278     #
1279     #
1280     #
1281     #
1282     #
1283     #
1284     #
1285     #
1286     #
1287     #
1288     #
1289     #
1290     #
1291     #
1292     #
1293     #
1294     #
1295     #
1296     #
1297     #
1298     #
1299     #
1300     #
1301     #
1302     #
1303     #
1304     #
1305     #
1306     #
1307     #
1308     #
1309     #
1310     #
1311     #
1312     #
1313     #
1314     #
1315     #
1316     #
1317     #
1318     #
1319     #
1320     #
1321     #
1322     #
1323     #
1324     #
1325     #
1326     #
1327     #
1328     #
1329     #
1330     #
1331     #
1332     #
1333     #
1334     #
1335     #
1336     #
1337     #
1338     #
1339     #
1340     #
1341     #
1342     #
1343     #
1344     #
1345     #
1346     #
1347     #
1348     #
1349     #
1350     #
1351     #
1352     #
1353     #
1354     #
1355     #
1356     #
1357     #
1358     #
1359     #
1360     #
1361     #
1362     #
1363     #
1364     #
1365     #
1366     #
1367     #
1368     #
1369     #
1370     #
1371     #
1372     #
1373     #
1374     #
1375     #
1376     #
1377     #
1378     #
1379     #
1380     #
1381     #
1382     #
1383     #
1384     #
1385     #
1386     #
1387     #
1388     #
1389     #
1390     #
1391     #
1392     #
1393     #
1394     #
1395     #
1396     #
1397     #
1398     #
1399     #
1400     #
1401     #
1402     #
1403     #
1404     #
1405     #
1406     #
1407     #
1408     #
1409     #
1410     #
1411     #
1412     #
1413     #
1414     #
1415     #
1416     #
1417     #
1418     #
1419     #
1420     #
1421     #
1422     #
1423     #
1424     #
1425     #
1426     #
1427     #
1428     #
1429     #
1430     #
1431     #
1432     #
1433     #
1434     #
1435     #
1436     #
1437     #
1438     #
1439     #
1440     #
1441     #
1442     #
1443     #
1444     #
1445     #
1446     #
1447     #
1448     #
1449     #
1450     #
1451     #
1452     #
1453     #
1454     #
1455     #
1456     #
1457     #
1458     #
1459     #
1460     #
1461     #
1462     #
1463     #
1464     #
1465     #
1466     #
1467     #
1468     #
1469     #
1470     #
1471     #
1472     #
1473     #
1474     #
1475     #
1476     #
1477     #
1478     #
1479     #
1480     #
1481     #
1482     #
1483     #
1484     #
1485     #
1486     #
1487     #
1488     #
1489     #
1490     #
1491     #
1492     #
1493     #
1494     #
1495     #
1496     #
1497     #
1498     #
1499     #
1500     #
1501     #
1502     #
1503     #
1504     #
1505     #
1506     #
1507     #
1508     #
1509     #
1510     #
1511     #
1512     #
1513     #
1514     #
1515     #
1516     #
1517     #
1518     #
1519     #
1520     #
1521     #
1522     #
1523     #
1524     #
1525     #
1526     #
1527     #
1528     #
1529     #
1530     #
1531     #
1532     #
1533     #
1534     #
1535     #
1536     #
1537     #
1538     #
1539     #
1540     #
1541     #
1542     #
1543     #
1544     #
1545     #
1546     #
1547     #
1548     #
1549     #
1550     #
1551     #
1552     #
1553     #
1554     #
1555     #
1556     #
1557     #
1558     #
1559     #
1560     #
1561     #
1562     #
1563     #
1564     #
1565     #
1566     #
1567     #
1568     #
1569     #
1570     #
1571     #
1572     #
1573     #
1574     #
1575     #
1576     #
1577     #
1578     #
1579     #
1580     #
1581     #
1582     #
1583     #
1584     #
1585     #
1586     #
1587     #
1588     #
1589     #
1590     #
1591     #
1592     #
1593     #
1594     #
1595     #
1596     #
1597     #
1598     #
1599     #
1600     #
1601     #
1602     #
1603     #
1604     #
1605     #
1606     #
1607     #
1608     #
1609     #
1610     #
1611     #
1612     #
1613     #
1614     #
1615     #
1616     #
1617     #
1618     #
1619     #
1620     #
1621     #
1622     #
1623     #
1624     #
1625     #
1626     #
1627     #
1628     #
1629     #
1630     #
1631     #
1632     #
1633     #
1634     #
1635     #
1636     #
1637     #
1638     #
1639     #
1640     #
1641     #
1642     #
1643     #
1644     #
1645     #
1646     #
1647     #
1648     #
1649     #
1650     #
1651     #
1652     #
1653     #
1654     #
1655     #
1656     #
1657     #
1658     #
1659     #
1660
```

F:\Development\PR\PR1\hbsp-extractOrder-WORKING\procOrder.pl
 Printed at 19:28 on 06 Feb 1999

```

1167 ## DETECT DIFFERENCE IN EXTRACTION STACK AND MAX EXTRACTION STACK
1168 if ($exit && (scalar(@singlestack) != scalar(@testsinglestack)) ) {
1169     my $t = '.vfyTok';
1170     my $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t : $eventRequiresFUP ; $fupcount++; $excpcount++; $eventDesc
1171     = " \<" . $thisRecordID . "\> checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING / INSPECT IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $eventType
1172     , $eventDesc, $guser, $LGDEBUG_FTN);
1173 }
1174 }
1175 my $FIND_ADD_CHANGE = 0;
1176 foreach(@addressesChangeIndicators) { s/[\n\r]//g; next if($ _ !~ /\s+?/); if($thisPreExtractionLine =~ /$ _/) { $FIND_ADD_CHANGE = 1; } }
1177 }
1178 if ($FIND_ADD_CHANGE) {
1179     my $t = '.chkact';
1180     my $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t : $eventRequiresFUP ; $fupcount++; $excpcount++; $eventDesc
1181     = " \<" . $thisRecordID . "\> checkMultiBuy_Extract :: POSSIBLE ADD/CHANGE REQUEST DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $eventType,
1182     $eventDesc, $guser, $LGDEBUG_FTN);
1183 }
1184 }
1185 my $FIND_QUESTION = 0;
1186 if($thisPreExtractionLine =~ /\?$/s/) { $FIND_QUESTION = 1; }
1187 if ($FIND_QUESTION) {
1188     my $t = '.chkqst';
1189     my $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t : $eventRequiresFUP ; $fupcount++; $excpcount++; $eventDesc
1190     = " \<" . $thisRecordID . "\> checkMultiBuy_Extract :: POSSIBLE ADD/CHANGE OR CARE REQUEST DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $ev
1191     entType, $eventDesc, $guser, $LGDEBUG_FTN);
1192 }
1193 }
1194 my $FIND_AI_ONLY = 0;
1195 if ($exit && scalar(@singlestack) == 2 && scalar(@testsinglestack) == 2) {
1196     ## $x = scalar(@singlestack);
1197     ## $y = scalar(@testsinglestack);
1198     ##print "-- debug -- AI TEST --- singlestackLen = $x --- testsinglestackLen = $y \n";
1199     if (grep /(A)|(T)/, @singlestack) { $FIND_AI_ONLY = 1; }
1200 }
1201 if ($FIND_AI_ONLY) {
1202     my $t = '.vfyTok';
1203     my $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t : $eventRequiresFUP ; $fupcount++; $excpcount++; $eventDesc
1204     = " \<" . $thisRecordID . "\> checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING: \"A\" or \"T\" ONLY DETECTED"; &logEvent(\*FH_log, $eventDesc
1205     , $eventDesc, $guser, $LGDEBUG_FTN);
1206 }
1207 }
1208 my $LIKELY_PARSE_ERRORS = 0;
1209 if ($exit && $thisPreExtractionLine =~ /[([)]+$/) { $LIKELY_PARSE_ERRORS = 1; }
1210 if ($LIKELY_PARSE_ERRORS) {
1211     my $t = '.vfyTok';
1212     my $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t : $eventRequiresFUP ; $fupcount++; $excpcount++; $eventDesc
1213     = " \<" . $thisRecordID . "\> checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING: BRACKETS FOUND IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $ev
1214     entType, $eventDesc, $guser, $LGDEBUG_FTN);
1215 }
1216 }
1217 }
1218 }
1219 }
1220 }
1221 }
1222 }
1223 }
1224 }
1225 }

```

Development PRJ:PRJ:dbasp_extractOrder - WORKING on 02 Feb 1999
Printed at 19:48 on 06 Feb 1999

```

1226     }
1227
1228
1229     ##print "\n-$thisrecordID- debug singlestack -- \n";
1230     ##print @singlestack;
1231     ##print "\n-- debug testsinglestack -- \n";
1232     ##print @testsinglestack;
1233     ##print "\n-$thisrecordID-----\n";
1234
1235
1236     return $exit;
1237 }
1238
1239
1240 # -----
1241 #
1242 #
1243
1244 sub detectBuyTokens {
1245     my($MULTI_BUY_ON,$Subj_buyToken,$thissubject,$thismailingID,$thisbodyTop,$ptrHASHLAYOUT,$ptrSCALARTopLines,$ptrSCALARbottomLines,$GDEBUG)
1246     = @_;
1247
1248     my %LAYOUT
1249     = %ptrHASHLAYOUT;
1250     my @topLines
1251     = @ptrSCALARTopLines;
1252     my @bottomLines
1253     = @ptrSCALARbottomLines;
1254     my $buyTokenPosition
1255     = 'none';
1256     $thisbuyStatus
1257     = undef;
1258     $thisitemsSelected
1259     = undef;
1260     $thisitemsSelectedstack
1261     = undef;
1262     my $exit = 0;
1263
1264     ## print "--debug-- buyToken= |$Subj_buyToken|\n";
1265
1266     FIND_BUY_TOKENS: {
1267         1261
1262         ## &checkMultiBuy_Extract returns success on a good match/extraction;
1263         ## @thisitemsSelected is created/populated at this time as well
1264         ##
1265         ##
1266         ## check $thisSubject
1267         ##
1268         ## print "\n\n=====debug-- SUBJECT = |$thissubject| \n";
1269         if(!($MULTI_BUY_ON && $thissubject =~ /$Subj_buyToken/) {
1270             $buyTokenPosition = ' s ' ; $thisbuyStatus = ('<'. $buyTokenPosition.'> '|'. $thissubject.'|');
1271         }
1272
1273         $buyTokenPosition = ' s ' ; $thisbuyStatus = ('<'. $buyTokenPosition.'> '|'. $thissubject.'|');
1274
1275         ## print "--debug-- thisbuyStatus= |$thisbuyStatus| \n";
1276         last FIND_BUY_TOKENS;
1277     }
1278
1279     if($MULTI_BUY_ON) {
1280         if(&checkMultiBuy_Extract($thissubject,$thismailingID,' s ',%LAYOUT,$GDEBUG)) {
1281             $buyTokenPosition = ' s ' ; $thisbuyStatus = ('<'. $buyTokenPosition.'> '|'. $thissubject.'|');
1282             last FIND_BUY_TOKENS;
1283         }
1284     }
1285
1286     ## check @topLines
1287     ## print "\n\n=====debug-- thisbodyTop = |$thisbodyTop| \n";
1288
1289
1290
1291
1292
1293

```

F:\Development\PRJ\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
printed at 19:48 on 06 Feb 1999

```

1294 if(!${MULTI_BUY_ON}) {
1295     foreach(@topLines) {
1296         $thisTopLine = $_;
1297         if($thisTopLine =~ /$gsubj_buyToken/) {
1298             $buyTokenPosition = 'T';
1299             $thisBuyStatus = 'L'. $buyTokenPosition. '> '. $thisTopLine;
1300             ## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";
1301         }
1302     }
1303     last FIND_BUY_TOKENS;
1304 }
1305 if($MULTI_BUY_ON) {
1306     if(&checkMultiBuy_Extract($thisBodyTop, $thismailingID, 'T', \%LAYOUT, $GDEBUG)) {
1307         $exit = 1;
1308         $buyTokenPosition = 'T';
1309         $thisBuyStatus = 'L'. $buyTokenPosition. '> ';
1310         last FIND_BUY_TOKENS;
1311     }
1312 }
1313 ## check @bottomLines
1314 if(!${MULTI_BUY_ON}) {
1315     foreach(@bottomLines) {
1316         $thisBottomLine = $_;
1317         if($thisBottomLine =~ /$gsubj_buyToken/) {
1318             $buyTokenPosition = 'B';
1319             $thisBuyStatus = 'L'. $buyTokenPosition. '> '. $thisBottomLine;
1320             ## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";
1321         }
1322     }
1323     last FIND_BUY_TOKENS;
1324 }
1325 if($MULTI_BUY_ON) {
1326     if(&checkMultiBuy_Extract($thisBodyBottom, $thismailingID, 'B', \%LAYOUT, $GDEBUG)) {
1327         $exit = 1;
1328         $buyTokenPosition = 'B';
1329         $thisBuyStatus = 'L'. $buyTokenPosition. '> ';
1330         last FIND_BUY_TOKENS;
1331     }
1332 }
1333 }
1334 }
1335 }
1336 }
1337 }
1338 }
1339 }
1340 }
1341 }
1342 }
1343 }
1344 }
1345 }
1346 }
1347 }
1348 }
1349 }
1350 }
1351 }
1352 }
1353 }
1354 }
1355 }
1356 }
1357 }
1358 }
1359 }
1360 }
1361 }
1362 }

```

F:\Development\PRJ\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
 Printed at 19:48 on 06 Feb 1999

```

1363         last FIND_BUY_TOKENS;
1364     }
1365 }
1366 }
1367 }
1368 }
1369 # print "--debug-- === $i ===== buyTokenPosition= {$buyTokenPosition}\n";
1370 # print "--debug-- === $i ===== buyTokenPosition= {$buyTokenPosition}\n";
1371 return $buyTokenPosition;
1372 }
1373 }
1374 }
1375 # -----
1376 # -----
1377 # -----
1378 # -----
1379 sub extractDedupItemsSelectedStack {
1380     my(@thisItemsSelected) = @_;
1381     my $thisItemsSelectedStack = undef;
1382     my $thisItemsSelectedStackSeparator = ',';
1383     my $i = undef;
1384     for($i = 0; $i <= $#thisItemsSelected; $i++) {
1385         my $item = $thisItemsSelected[$i]; # all items must be normalized!
1386         next if($item !~ /\s+?/);
1387         $item=&normalizeSpacing($item);
1388         $thisItemsSelectedStack .= ($item.$thisItemsSelectedStackSeparator);
1389     }
1390     return $thisItemsSelectedStack;
1391 }
1392 }
1393 }
1394 }
1395 }
1396 }
1397 }
1398 }
1399 $thisItemsSelectedStack =~ s/\\s+?//;
1400 }
1401 return $thisItemsSelectedStack;
1402 }
1403 }
1404 # -----
1405 # -----
1406 # -----
1407 }
1408 sub getDateStamp {
1409     # get standardized date stamp (UNIX, WINNT)
1410     # returns a date stamp
1411     my($platform, $GDEBUG_FTN) = @_;
1412     if(!defined($platform)) { $platform = 'WIN'; }
1413     my $GTIME = '';
1414     my $day = undef;
1415     my $val = undef;
1416     if ($platform ne 'WIN') { $val = 'date '+%m/%d/%Y-%a'; }
1417     else {
1418         chop($val);
1419         $val =~ s/(\w+)(\s+)(\d+)(\s+)(\d+)/(\d{2})(\d{2})/;
1420         $val = $val.$day.$val;
1421     }
1422     chop($val);
1423     $val = $val.$day.$val;
1424     $val = $val.$day.$val;
1425     $val = $val.$day.$val;
1426     $val = $val.$day.$val;
1427     if ($platform eq 'WIN') {
1428         $GTIME = 'echo "' . time . '"';
1429         $GTIME =~ s/\\s+?//;
1430     }
1431 }

```

F:\Development\PR\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
 Printed at 19:48 on 06 Feb 1999

```

1432 my $leadingDigit = $1;
1433 if (length($leadingDigit) == 1) { $leadingDigit = ('0' . $leadingDigit); }
1434 $val .= $leadingDigit.$2.$3.$4.'-'. $day;
1435 }
1436
1437 if($LDEBUG_FTN) { print "DEBUG_gdate=".$val."\n"; }
1438
1439 return $val;
1440 }
1441 }
1442 }
1443 # -----
1444 #
1445 #
1446 sub getFieldKeys {
1447   my($ptrLayout) = @_;
1448   my(%LAYOUT) = %$ptrLayout;
1449
1450   ## prepare and count required fields from body data layout for check against body
1451
1452   my(@fieldkeys) = undef;
1453   my($k)
1454   foreach $k (sort keys %LAYOUT) {
1455     my $f = scalar(@fieldkeys); print "---debug-- scalarStartFtn=$f \n";
1456     if ($k =~ /\S+?/) { push(@fieldkeys,$k); }
1457   }
1458   shift @fieldkeys;
1459   ## $f = scalar(@fieldkeys); print "---debug-- scalarEndFtn=$f \n";
1460   return @fieldkeys;
1461 }
1462
1463 # -----
1464 #
1465 #
1466 sub getKeyPos_Outfile {
1467   my($ptrHASHrecordTemplate) = @_;
1468   my(%HASHRecordTemplate) = %$ptrHASHrecordTemplate;
1469   my @record
1470   = undef;
1471   $salutationFieldPos = -1;
1472   $IDFieldPos = -1;
1473   $mailingIDFieldPos = -1;
1474   $firstNameFieldPos = -1;
1475   $lastNameFieldPos = -1;
1476   $titleFieldPos = -1;
1477   $stateFieldPos = -1;
1478   $countryFieldPos = -1;
1479   $postalCodeFieldPos = -1;
1480   $emailFieldPos = -1;
1481   $companyFieldPos = -1;
1482   $addressFieldPos = -1;
1483   $departmentFieldPos = -1;
1484   $cityFieldPos = -1;
1485   $fromAddressFieldPos = -1;

```


F:\Development\PRJ\PRJ_hbsp_extractOrder-WORKING\GprocOrder.pl
 Printed at 19:48 on 06 Feb 1999

```

1501 my $i
1502         = 0;
1503 @record = sort(keys(%HASHrecordTemplate));
1504
1505 #print "----DEBUG---"; print @record; sleep 4;
1506
1507 for($i=0; $i <= $#record; $i++) {
1508     $record[$i] = uc($record[$i]);
1509
1510     CASE: {
1511         if ($record{$i} == /FROM_ADDRESS/)
1512             if ($record{$i} == /(EMAIL)|(E-MAIL)|(INTERNET_ADOR)|(INTERNET_AD)|(INTERNETAD)/)
1513                 $fromAddressFieldPos = $i; last CASE;
1514             if ($record{$i} == /(COUNTRY)|(CTRY)/)
1515                 $countryFieldPos = $i; last CASE;
1516             if ($record{$i} == /(ZIP)|(POSTAL)/)
1517                 $postalCodeFieldPos = $i; last CASE;
1518             if ($record{$i} == /(INST)|(COMPANY)|(ORGANIZATION)/)
1519                 $companyFieldPos = $i; last CASE;
1520             if ($record{$i} == /(DEPARTMENT)/)
1521                 $departmentFieldPos = $i; last CASE;
1522             if ($record{$i} == /(ADDR)|(ADDRESS)/)
1523                 $addressFieldPos = $i; last CASE;
1524             if ($record{$i} == /(CITY)/)
1525                 $cityFieldPos = $i; last CASE;
1526             if ($record{$i} == /(CUSTNO)/)
1527                 $IDFieldPos = $i; last CASE;
1528             if ($record{$i} == /(LISTID)/)
1529                 $mailingIDFieldPos = $i; last CASE;
1530             if ($record{$i} == /(PRIORITY)|(MAILING)/)
1531                 $firstNameFieldPos = $i; last CASE;
1532             if ($record{$i} == /(.*F+W+NAME.*)|(FIRST)/)
1533                 $lastNameFieldPos = $i; last CASE;
1534             if ($record{$i} == /(.*L+W+NAME.*)|(LAST)/)
1535                 $IXFieldPos = $i; last CASE;
1536             if ($record{$i} == /(.*IX.*)/)
1537                 $titleFieldPos = $i; last CASE;
1538             if ($record{$i} == /STATE/ || $record{$i} == /\A_CS/)
1539                 $stateFieldPos = $i; last CASE;
1540         }
1541     }
1542
1543     if ($firstNameFieldPos == -1)
1544         print STDOUT "[HEADER] WARNING: \\\w+\\f\\w+NAME\\ was not found!\\n";
1545     if ($lastNameFieldPos == -1)
1546         print STDOUT "[HEADER] WARNING: \\\w+\\L\\w+NAME\\ was not found!\\n";
1547     if ($titleFieldPos == -1)
1548         print STDOUT "[HEADER] WARNING: \\\w+\\IX\\w+NAME\\ was not found!\\n";
1549     if ($stateFieldPos == -1)
1550         print STDOUT "[HEADER] WARNING: \\\w+\\STATE\\ was not found!\\n";
1551     if ($postalCodeFieldPos == -1)
1552         print STDOUT "[HEADER] WARNING: \\\w+\\ZIP\\POSTAL\\ was not found!\\n";
1553     if ($countryFieldPos == -1)
1554         print STDOUT "[HEADER] WARNING: \\\w+\\COUNTRY\\(CTRY)\\ was not found!\\n";
1555     if ($emailFieldPos == -1)
1556         print STDOUT "[HEADER] WARNING: \\\w+\\EMAIL\\(E-MAIL)\\(INTERNET_ADOR)\\(INTERNET_AD)\\ was not found!\\n";
1557
1558     sub getNextDBkey_return {
1559         my($keyFileName,$LGDBUG_FTN) = @_;
1560
1561         $LOCK_SH = 1;
1562         $LOCK_EX = 2;
1563         $LOCK_NB = 4;
1564         $LOCK_UN = 8;
1565         $POS_BOF = 0;
1566         $POS_CUR = 1;
1567         $POS_EOF = 2;
1568
1569         $/_ = undef;
1570         $startKeys = 0;
1571         if(! -s $keyFileName) { echo $startKeys >> $keyFileName; }
1572         OPEN_DB: if (open(KEYS, ">+< $keyFileName")) {exit = 0; $eventType = 'FAIL'; $eventDesc = "getNextDBkey_return::Cannot open: \\\$keyFileName\\"; &logEven
1573             t(\\*FH_log, $eventType, $eventDesc, $Guser, $LGDBUG_FTN);}
1574     }

```

F:\Development\PRJ\PRJ hbsp extractOrder-WORKING\procOrder.pl
 Printed at 19:48 on 06 Feb 1999

```

1568 flock(KEYS, $LOCK_EX);
1569 my($count) = <KEYS>;
1570 $nextkey = $count+1;
1571 seek(KEYS,0,$POS_EOF);
1572 print KEYS $nextkey;
1573
1574 flock(KEYS, $LOCK_UN);
1575 close(KEYS);
1576 return $nextkey;
1577
1578 }
1579
1580 # -----
1581 #
1582 #
1583 #
1584 sub HASHloadFieldMap {
1585   my($fileName,$LGDEBUG_FTN) = @_;
1586   if (open(FH,"$fileName")) { $eventType = 'FAIL'; $eventDesc = "HASHloadFieldMap::Cannot open: \[$keyFileName \]"; &logEvent(\*FH_log, $eventType, $event
1587     Desc, $user, $LGDEBUG_FTN);
1588   }
1589   my $buf = "\n";
1590   my(@fieldMapTemp) = <FH>;
1591   my($key) = undef;
1592   my($value) = undef;
1593   my(%fieldMap) = undef;
1594   while ($key) {
1595     my($key,$value) = split(/[\t\r\n]/,$item);
1596     $value =~ s/[\n\r]//g;
1597     $fieldMap{$key} = $value;
1598   }
1599   close(FH);
1600   foreach (@fieldMapTemp) {
1601     my $item = $_;
1602     next if(($item =~ /\s+/) || ($item =~ /\s+?/)); # comments
1603     ($key,$value) = split(/[\t\r\n]/,$item);
1604     $value =~ s/[\n\r]//g;
1605     $fieldMap{$key} = $value;
1606   }
1607   return %fieldMap;
1608 }
1609
1610 # -----
1611 #
1612 #
1613 #
1614 sub loadFile_SCALAR {
1615   my($fileName) = @_;
1616   if (open(FILE,"$fileName")) { $eventType = 'FAIL'; $eventDesc = "loadArray::Cannot open fileName: \[$fileName \]"; &logEvent(\*FH_log, $eventType, $eventD
1617     esc, $user, $LGDEBUG_FTN);
1618   }
1619   my $buf = "\n";
1620   while ($buf) {
1621     my($key,$value) = undef;
1622     while(<FILE>) {
1623       my $buf = $_;
1624       next if($buf !~ /\s+?/ || $buf =~ /\s+?/);
1625     }
1626   }
1627 }
1628
1629 my($all) = undef;
1630 while(<FILE>) {
1631   my $buf = $_;
1632   next if($buf !~ /\s+?/ || $buf =~ /\s+?/);
1633 }
1634

```

F:\Development\PRJ\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
 Printed at 16:48 on 06 Feb 1999

```

1635     push(@all, $buf);
1636 }
1637 }
1638 return @all;
1639 }
1640 }
1641 }
1642 # -----
1643 # -----
1644 # -----
1645 sub logEvent {
1646     my($FH,$eventType,$eventDesc,$user,$LGDEBUG_FTN) = @_;
1647     my($FH,$eventType,$eventDesc,$user,$LGDEBUG_FTN) = @_;
1648     my $dateNow = &getDateStamp('WIN',$LGDEBUG_FTN);
1649     my $printEvent = STDOUT;
1650     if($eventType eq 'ABORT') {
1651         $foo = "mapisend -u administrator -p password -r aestes\@e-dialog.com -s *** $dateNow procOrder - Critical Error" -m "$Guser $GclientName $eventT
1652         ype $eventDesc";
1653         die "Error! Cannot log! ". $eventType."\\t". $eventDesc."\\n";
1654     }
1655     else { $eventType =~ s/_X_//; }
1656     # write event to activity log
1657     &writeRecord(*FH_log,$GfieldDelimiter,$thisLogKey=&getNextDBkey_return($GactivityLogKeyName,$LGDEBUG_FTN),\%activityLogRecord,'',$LGDEBUG_F
1658     TN);
1659     if($eventType eq 'FAIL') {
1660         $foo = "mapisend -u administrator -p password -r aestes\@e-dialog.com -s *** $dateNow procOrder - Critical Error" -m "$Guser $GclientName
1661         $eventType $eventDesc";
1662         exit;
1663     }
1664 }
1665 }
1666 }
1667 }
1668 }
1669 }
1670 }
1671 }
1672 }
1673 }
1674 }
1675 }
1676 }
1677 }
1678 sub normalizeSpacing {
1679     my($dirty) = @_;
1680     $dirty =~ s/\s{2,}/space/g; $dirty =~ s/\s+//; $dirty =~ s/\s+//;
1681     $clean = $dirty;
1682     return $clean;
1683 }
1684 }
1685 }
1686 }
1687 }
1688 }
1689 }
1690 }
1691 }
1692 }
1693 }
1694 }
1695 }
1696 }
1697 }
1698 }
1699 }
1700 }

```

F:\Development\PRJ\PRJ_hbsp_extractor\order-WORKING\order.pl
 Printed at 19:48 on 06 Feb 1999

```

1701 my $Errors = undef;
1702 my @fieldNames = undef;
1703 my $PID = -1;
1704 my $RecCount = 0;
1705 my $EXIT = 1;
1706
1707 # construct/open DSN
1708
1709 my $Conn = undef;
1710 $Conn = new Win32::OLE("ADODB.Connection");
1711 $Conn->open($LAYOUT{'thisMailingDSN'}, "", "");
1712
1713 my(%DB) = undef;
1714
1715 # Load data into recordset
1716
1717 my $mailingTable = $LAYOUT{'thisMailingTable'};
1718 my %foo = undef;
1719
1720 # Look to see if update record exists, $thisQueryKey
1721
1722 unless($RECORD_RECOVERED) {
1723     my $RECORD_EXISTS = 1;
1724     my $foo = &querySELECT_DB($LAYOUT, $thisQueryFieldName, $thisQueryKey, $LGDEBUG_FTN);
1725     if($foo{'_exit_'} < 1) { $RECORD_EXISTS = 0; $EXIT = 0; return $EXIT; }
1726 }
1727
1728 $thisQuery = "UPDATE $mailingTable SET $fieldName = '$fieldValue' WHERE $thisQueryFieldName = '$thisQueryKey'";
1729
1730 ";
1731
1732 $RS = $Conn->Execute($thisQuery);
1733
1734 ##$foo = %$RS;
1735 foreach(keys(%foo)) { print "---key = $_ --- value = $foo{$_} \n"; }
1736
1737 if(!($RS)) {
1738     $Errors = $Conn->Errors();
1739     print "Errors:\n";
1740     foreach $error (keys %$Errors) {
1741         print $error->{Description}, "\n";
1742     }
1743     $eventType = 'WRN_FUP'; $eventRequiresFUP .= (($eventRequiresFUP =~ /badMDB/) ? '' : '.badMDB'); $eventDesc = "\<" . $thisRecordID. ">" . ($LAYOUT{'thisMailingDSN'}) queryUPDATE_DB :: Cannot create RS: \[$thisQuery\]; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
1744     UT{'thisMailingDSN'};
1745     $EXIT = 0;
1746     return $EXIT;
1747 }
1748
1749 $eventType = 'INFO'; $eventDesc = "\<" . $thisRecordID. ">" . ($LAYOUT{'thisMailingDSN'}) queryUPDATE_DB :: UPDATE SUCCESSFUL: \[$thisQuery\]; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
1750
1751 return $EXIT;
1752 }
1753
1754 #
1755 # -----
1756 #
1757 #
1758
1759 sub querySELECT_DB {
1760     my($ptrLAYOUT, $thisQueryFieldName, $thisQueryKey, $LGDEBUG_FTN) = @_;
1761     #print "*** DSN = $LAYOUT{'thisMailingDSN'} \n";
1762     my($LAYOUT) = %$ptrLAYOUT;
1763     my $RS = undef;
1764
1765     my $RS = undef;
1766
1767

```

F:\Development\PRJ\PRJ_hbsp_extractOrder-WORKING\GprocOrder.pl
Printed at 19:48 on 06 Feb 1999

```

1767 my $Conn = undef;
1768 my $Errors = undef;
1769 my @fieldNames = undef;
1770 my $PID = -1;
1771 my $recCount = 0;
1772
1773 # construct/open DSN
1774
1775 my $Conn = undef;
1776 $Conn = new Win32::OLE("ADODB.Connection");
1777 $Conn->open($LAYOUT{'thismailingDSN'}, "", "");
1778
1779 my($ODB) = undef;
1780 $ODB{'_exit_'} = 1;
1781
1782 # Load data into recordset
1783
1784 my $mailingTable = $LAYOUT{'thismailingTable'};
1785
1786 $thisQuery = "SELECT * FROM $mailingTable WHERE $thisQueryFieldName = \"'$thisQueryKey'\"";
1787
1788 $RS = $Conn->Execute($thisQuery);
1789
1790 if(!$RS) {
1791     $Errors = $Conn->Errors();
1792     print "Errors:\n";
1793     foreach $error (keys %$Errors) {
1794         print $error->{Description}, "\n";
1795     }
1796     $eventType = 'FAIL'; $eventDesc = " \<" . $thisRecordID . " \> ($LAYOUT{'thismailingDSN'}) querySELECT_DB :: Cannot create RS: \"'$thisQuery'\""; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
1797 }
1798
1799 }
1800
1801 my $i = undef;
1802
1803 for($i=0; $i < $RS->Fields->Count; $i++) { $fieldNames[$i] = $RS->Fields->Item($i)->Name; }
1804
1805 my $RESULT_FLAG = 1;
1806
1807 if($RS->EOF) { $RESULT_FLAG = 0; }
1808 $RS -> MoveNext;
1809
1810 if(!$RS->EOF) { $RESULT_FLAG = -1; }
1811 $RS -> MovePrevious;
1812
1813 if($RESULT_FLAG < 1) {
1814     ((($eventRequiresFUP =~ /badMDB/)?'':'badMDB')); $FUPCount++; $eventType = 'ERR_FUP'; $eventDesc = " \<" . $thisRecordID . " \> ($LAYOUT{'thismailingDSN'}) querySELECT_DB :: Bad Lookup ($RESULT_FLAG): \"'$thisQuery'\""; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
1815 }
1816
1817 $ODB{'_exit_'} = $RESULT_FLAG;
1818
1819 return %ODB;
1820 }
1821
1822 while ( !$RS->EOF ) {
1823     for($i=0; $i < $RS ->Fields->Count; $i++) {
1824         $ODB{$fieldNames[$i]} = uc($RS->Fields->Item($i)->value);
1825     }
1826     $RS->MoveNext; $recCount++;
1827 }
1828
1829 }
1830 }
1831
1832 $Conn->Close;
1833

```

F:\Development\PR\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
 Printed at 19:48 on 06 Feb 1999

```

1834 return %DB;
1835 }
1836
1837 # -----
1838 #
1839 #
1840
1841 sub read_DSNTAOAH {
1842
1843   my($DSN, $LGDEBUG_FTN) = @_;
1844   ##print "** DSN = $DSN \n";
1845
1846   $RS = undef;
1847   $Conn = undef;
1848   $Errors = undef;
1849   @fieldNames = undef;
1850   $PID = -1;
1851   $reccount = 0;
1852
1853   # construct/open DSN
1854
1855   $Conn = new Win32::OLE("ADODB.Connection");
1856   $Conn->Open($DSN, "", "");
1857   my(@FILE) = undef;
1858
1859   # Load data into recordset
1860
1861   $thisQuery = "SELECT * FROM Email";
1862   $RS = $Conn->Execute($thisQuery);
1863
1864   if(!$RS) {
1865     $Errors = $Conn->Errors();
1866     print "Errors:\n";
1867     foreach $error (keys %$Errors) {
1868       print $error->{description}, "\n";
1869     }
1870     $eventType = 'FAIL'; $eventDesc = "read_DSNTAOAH: Cannot create RS: \"$thisQuery\""; &logEvent(\*FH_log, $eventType, $eventDesc, $user, $LGDEBUG_FTN);
1871   }
1872
1873   my $i = undef;
1874
1875   for($i=0; $i < $RS->Fields->Count; $i++) { $fieldNames[$i] = $RS->Fields->Item($i)->Name; }
1876
1877   while ( !$RS->EOF ) {
1878     for($i=0; $i < $RS->Fields->Count; $i++) {
1879       $FILE[$reccount][$fieldNames[$i]] = $RS->Fields->Item($i)->value;
1880     }
1881     $RS->moveNext; $reccount++; if ($reccount % 20 == 0) { print STDOUT "+"; }
1882   }
1883
1884   $Conn->Close;
1885
1886   return @FILE;
1887 }
1888
1889 # -----
1890 #
1891 #
1892 #
1893 #
1894 #
1895 #
1896 #
1897 #
1898
1899 sub removeTabs {
1900
1901   my($tabs) = @_;

```

F:\Development\PRJ\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
 Printed at 19:48 on 06 Feb 1999

```

1902 $tabs = s/\t//g;
1903
1904 return $tabless
1905
1906 }
1907
1908 # -----
1909 #
1910 #
1911 #
1912 sub subjectsynonymlookup {
1913
1914   my($subject,$ptrHASHLAYOOUT) = @_;
1915   my($LAYOOUT) = %$ptrHASHLAYOOUT;
1916   my($thismailingID) = -1;
1917
1918   $subject =~ /\s*([^\s\+\\-]{1,3})\s*/;
1919   $thisSubjectMailingIDToken = $1;
1920
1921   if(defined($LAYOOUT{$thisSubjectMailingIDToken})) { $thismailingID = $LAYOOUT{$thisSubjectMailingIDToken}; }
1922
1923   ## print "-- debug -- ATTEMPT MAILING ID RECOVERY... TOKEN= $thisSubjectMailingIDToken|... RESULT MAILINGID= $thismailingID\n";
1924
1925   return $thismailingID
1926 }
1927
1928 # -----
1929 #
1930 #
1931 #
1932 sub writerecord {
1933
1934   # takes FH by ref; has _e_ feature for values
1935
1936   my($FH,$Gfielddelimiter,$logKey,$ptrHASHrecord,$RECORD_TYPE,$LGDEBUG_FTN) = @_;
1937
1938   flock($FH, $LOCK_EX);
1939
1940   my $fileEmpty = 0;
1941   if(-z $FH) { $fileEmpty = 1; }
1942
1943   %HASHrecord = %$ptrHASHrecord;
1944
1945   $space = " ";
1946   $exit = 1;
1947
1948   $LOCK_SH = 1;
1949   $LOCK_EX = 2;
1950   $LOCK_NB = 4;
1951   $LOCK_UN = 8;
1952
1953   my $record = undef;
1954   my $header = undef;
1955   my $key = undef;
1956
1957   foreach $key(sort(keys(%HASHrecord))) {
1958     $value = $HASHrecord{$key};
1959     if($fileEmpty) { $key =~ s/[a-z]{1,2}_//; $header .= (uc($key).$Gfielddelimiter); }
1960     if($value =~ /\^_e_{2,}/) { $value = eval($1); }
1961
1962     $record .= ($value.$Gfielddelimiter);
1963
1964     # remove last field delimiter
1965   }
1966   chop($record);

```

C:\Development\PP_JPP\hisp_extractOrder-WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999

```
1971 if($fileEmpty) { chop($header); print $FH $header."\\n"; }
1972 if($ADD_CHANGE_ON && $RECORD_TYPE ne 'BODY') { $record = &canonicalize($record); }
1973 print $FH $record."\\n";
1974
1975 flock($FH, $LOCK_UN);
1976 }
1977 }
1978
1979 ---END---
1980
```


Claims

1 1. A machine-based method comprising
2 analyzing an e-mail message to derive response
3 information concerning a commercial transaction, and
4 based on the derived information, and
5 automatically generating commercial transaction data
6 in a format that is usable to automatically complete the
7 commercial transaction.

1 2. The method of claim 1 in which the commercial
2 transaction comprises an order for a product or service.

1 3. The method of claim 1 in which the e-mail
2 message comprises at least part of an e-mail sent to a
3 customer and responses of the customer to the e-mail.

1 4. The method of claim 1 in which the automatic
2 completion of the commercial transaction comprises order
3 fulfillment.

1 5. A machine-based method comprising
2 sending an e-mail message to a customer offering a
3 product or service for sale, the e-mail message comprising
4 locations for response by the customer indicating his
5 intention to order the product or service,
6 receiving from the customer an e-mail message that
7 includes the response,
8 based on the received e-mail, automatically
9 generating order information in a format usable
10 automatically by an order fulfillment system to cause the
11 order to be filled.

1 6. A machine-based method comprising
2 analyzing an e-mail message to derive response
3 information concerning a commercial transaction,
4 automatically identifying response information which
5 requires resolution of an issue with the source of the e-
6 mail message, and

7 automatically managing an e-mail dialog with the
8 source to resolve the issue.

1 7. The method of claim 6 in which at least some of
2 the e-mail dialog is performed automatically.

1 8. Software guided interactive e-mail dialogs to
2 resolve, on behalf of a vendor, customer issues that occur
3 in direct response e-mails that are automatically identified
4 as requiring a dialog.

1 9. A machine-based method comprising
2 automatically sorting e-mail messages, based on
3 response information contained in the messages, into e-mail
4 messages that can be processed automatically to generate
5 commercial transactions, e-mail messages in which the
6 response information is inadequate to permit generation of
7 commercial transactions, and e-mail messages that may be
8 subjected to exception handling to yield information that is
9 sufficient to generate commercial transactions.

1 10. A machine-based method comprising
2 analyzing an e-mail message to derive response
3 information concerning a commercial transaction, and
4 automatically generating a confirmatory e-mail
5 message to the source of the e-mail message confirming that
6 the commercial transaction has been or will be completed.

1 11. A machine-based method comprising
2 receiving inbound e-mail messages that result from
3 corresponding outbound e-mail messages associated with a
4 marketing program, the inbound messages containing response
5 information, each of the outbound messages being associated
6 with a distinct piece of the marketing program, and
7 automatically associating the response information
8 in each of the inbound messages with the corresponding
9 distinct piece of the marketing program.

1 12. The method of claim 11 in which the piece
2 comprises a marketing campaign or a marketing flight.

1 13. The method of claim 11 in which the inbound
2 messages contain information that links them to the
3 corresponding outbound messages, and the associating step
4 uses the link information.

1 14. The method of claim 13 further comprising
2 automatically parsing the inbound messages for order
3 information.

1 15. A machine-based method comprising
2 sending outbound e-mail messages associated with
3 commercial transactions,
4 storing information related to each of the outbound
5 messages in a database, the information being useful for
6 completing the commercial transactions, the information not
7 being contained in the outbound messages,
8 analyzing inbound e-mail messages that result from
9 the outbound messages and that contain response information
10 useful in completing the commercial transactions, and
11 automatically merging the response information with
12 corresponding information in the database for use in
13 completing the transactions.

1 16. A machine-based method comprising
2 sending outbound e-mail messages associated with
3 commercial transactions,
4 storing information related to each of the outbound
5 messages in a database, the information being useful for
6 completing the commercial transactions, the information not
7 being contained in the outbound messages,
8 analyzing inbound e-mail messages that result from
9 the outbound messages and that contain response information
10 useful in completing the commercial transactions,

11 identifying inbound e-mail messages that cannot be
12 processed automatically to generate the commercial
13 transactions, and
14 using the database information to assist in
15 exception handling of the identified inbound messages.

1/6

WILLIAM HERP

10

FROM: HARVARD BUSINESS SCHOOL PUBLISHING
SENT: WEDNESDAY, DECEMBER 16, 1998 9:19 PM
TO: WILLIAM HERP
SUBJECT: = NEW INSIGHTS FROM HARVARD BUSINESS REVIEW

HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION
BOSTON, MASSACHUSETTS USA

THURSDAY, DECEMBER 17, 1998

DEAR WILLIAM HERP,

ON THURSDAY, DECEMBER 3RD, WE WROTE YOU REGARDING A SPECIAL OFFER ON THE HARVARD BUSINESS REVIEW PAPERBACK SERIES. SINCE WE HAVE NOT HEARD BACK, WE WANTED TO FOLLOW UP BEFORE THIS SPECIAL OFFER CLOSES. IF YOU ARE SIMPLY NOT INTERESTED, WE APOLOGIZE FOR THE INTRUSION. BELOW PLEASE FIND THE ORIGINAL OFFER IN ITS ENTIRETY.

THE HARVARD BUSINESS REVIEW PAPERBACK SERIES BRINGS YOU THE LATEST AND MOST SIGNIFICANT THINKING ON TODAY'S MOST PRESSING MANAGEMENT CHALLENGES. THESE INSIGHTFUL COLLECTIONS ARE THE DEFINITIVE RESOURCE FOR PROFESSIONALS.

EACH TITLE:

- + PROVIDES A BROAD UNDERSTANDING OF AN ISSUE
- + IS CLEARLY WRITTEN AND, IN MANY CASES, DRAWS UPON REAL COMPANY EXAMPLES
- + HELPS YOU CONSTRUCT A USEFUL CONCEPTUAL FRAMEWORK FOR DECISION-MAKING AND IMPLEMENTATION
- + CONTAINS EIGHT ARTICLES FROM HARVARD BUSINESS REVIEW

◇ EACH PAPERBACK IS \$19.95 PLUS SHIPPING AND HANDLING ◇

TO ORDER ONE OR MORE OF THESE PAPERBACKS, SIMPLY REPLY TO THIS MESSAGE AND
NOTE THE LETTER (A-F) OF THE HARVARD BUSINESS REVIEW PAPERBACK YOU WOULD LIKE
TO RECEIVE. PLEASE TYPE THE LETTERS IN THE 1ST LINE OF THE BODY OF YOUR REPLY
E-MAIL. SHIPPING AND HANDLING CHARGES WILL BE APPLIED TO EACH ORDER.

YOUR CHOICES (DETAILED BELOW) ARE:

- A: HARVARD BUSINESS REVIEW ON CHANGE PAPERBACK
B: HARVARD BUSINESS REVIEW ON KNOWLEDGE MANAGEMENT PAPERBACK
C: HARVARD BUSINESS REVIEW ON STRATEGIES FOR GROWTH PAPERBACK
D: HARVARD BUSINESS REVIEW ON MEASURING CORPORATE PERFORMANCE PAPERBACK
E: HARVARD BUSINESS REVIEW ON LEADERSHIP PAPERBACK

F: THE EXECUTIVE COLLECTION - ALL FIVE TITLES FOR \$89

OR IF YOU PREFER, CALL 1-800-668-6780 (617-496-1449 OUTSIDE THE U.S.)
MON. - FRI. 8 A.M. - 6 P.M. EST. PLEASE BE SURE TO MENTION PRIORITY CODE 3202.

[illegible]

FIG. 1A

2/6

A: ** HARVARD BUSINESS REVIEW ON CHANGE **
 PROVIDES LANDMARK IDEAS TO HELP YOU UNDERSTAND THE BEST WAYS FOR YOUR
 ORGANIZATION TO MANAGE CHANGE. INCLUDES ARTICLES BY JOHN KOTTER AND MORE.
 (240 PP/#8842/\$19.95)

B. ** HARVARD BUSINESS REVIEW ON KNOWLEDGE MANAGEMENT **
HIGHLIGHTS THE LEADING-EDGE THINKING AND PRACTICAL APPLICATIONS ON HOW COMPANIES
GENERATE, COMMUNICATE, AND LEVERAGE KNOWLEDGE ASSETS. INCLUDES ARTICLES BY PETER
DRUCKER, JOHN SEELY BROWN, AND MORE.
(240 PP/#8818/\$19.95)

C: ** HARVARD BUSINESS REVIEW ON STRATEGIES FOR GROWTH **
PRESENTS THE LATEST TACTICS FOR HELPING MANAGERS FIND AND EXPLOIT THE BEST
OPPORTUNITIES FOR GROWTH AND PROFITABILITY. INCLUDES ARTICLES BY ARIE DE GEUS,
JEFFREY RAYPORT, AND MORE.
(240 PP/#8850/\$19.95)

D: ** HARVARD BUSINESS REVIEW ON MEASURING CORPORATE PERFORMANCE **
OFFERS INSIGHT ON WHAT YOU NEED TO MEASURE AND HOW PERFORMANCE MEASURES CAN
ALIGN AN ORGANIZATION AND BOOST PRODUCTIVITY. INCLUDES ARTICLES BY PETER DRUCKER,
ROBERT KAPLAN AND DAVID NORTON, AND MORE.
(240 PP/#8826/\$19.95)

E: ** HARVARD BUSINESS REVIEW ON LEADERSHIP **
PRESENTS PROVEN FUNDAMENTALS OF LEADERSHIP AND CHALLENGES MANY LONG-HELD
ASSUMPTIONS ABOUT THE TRUE SOURCES OF POWER AND AUTHORITY. INCLUDES ARTICLES BY
JOHN KOTTER, JOSEPH BADARACCO, JR., AND MORE.
(240 PP/#8834/\$19.95)

F: PURCHASE THE EXECUTIVE COLLECTION (INCLUDES ALL FIVE TITLES) FOR JUST \$89- A SAVINGS OF MORE THAN 10%. (PRODUCT #7419BN)

[illegible]

FIG. 1B

3/6

[illegible]

◇ EACH PAPERBACK IS \$19.95 PLUS SHIPPING AND HANDLING ◇

IMPORTANT NOTE FOR OUR CUSTOMERS OUTSIDE THE U.S.: PURCHASERS ARE RESPONSIBLE FOR ALL DUTIES, TAXES, BROKERAGE FEES, AND/OR IMPORT FEES IMPOSED BY THE COUNTRY OF IMPORT. SHIPPING AND HANDLING CHARGES WILL BE APPLIED TO YOUR ORDER. DELIVERY TO CANADA: \$14.00 FOR THE FIRST TITLE, \$2.00 FOR EACH ADDITIONAL TITLE. INTERNATIONAL DELIVERY OUTSIDE NORTH AMERICA: \$20.00 FOR THE FIRST TITLE, \$5.00 FOR EACH ADDITIONAL TITLE.

PLEASE ALSO REVIEW AND UPDATE THE ADDRESS INFORMATION BELOW SO THAT WE CAN
PROCESS YOUR REQUEST PROMPTLY.

FIRST NAME: [WILLIAM]
LAST NAME: [HERP]
TITLE: [PRESIDENT]
COMPANY: [E-CARE GROUP INC.]
DEPARTMENT: []
ADDRESS1: [1646 MASSACHUSETTS AVE.]
ADDRESS2: []
ADDRESS3: []

CITY: [LEXINGTON]
PROVINCE/STATE: [MA]
POSTAL/ZIP CODE: [2173]
COUNTRY: []
PHONE: []
FAX: []
EMAIL: [WHERP@E-CARE.COM]

[[891270||3202|]]

FIG. 1C

4/6

WILLIAM HERP

FROM: HARVARD BUSINESS SCHOOL PUBLISHING
SENT: MONDAY, FEBRUARY 08, 1999 8:25 PM
TO: WILLIAM HERP
SUBJECT: ** A FREE NO-OBLIGATION TRIAL FROM HARVARD BUSINESS REVIEW

FROM THE DESK OF LAURA WINIG
HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION
BOSTON, MASSACHUSETTS
MONDAY, FEBRUARY 8, 1999

~INTRODUCING~

BENCHMARKING A NEW THREE-PART VIDEO SERIES FROM HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION

TO TAKE *BENCHMARKING* FOR A NO-OBLIGATION 14-DAY TEST DRIVE, SIMPLY REPLY TO THIS E-MAIL WITH THE WORD "YES" IN THE SUBJECT LINE

DEAR WILLIAM HERP:

INTERESTED IN UNEARTHING NEW IDEAS AND UNCONVENTIONAL SOLUTIONS FOR THE CHALLENGES FACING YOUR COMPANY? HERE AT THE PUBLISHING ARM OF HARVARD BUSINESS SCHOOL, WE'VE CREATED AN EXCITING NEW PROGRAM THAT CAN SHOW YOU HOW SOME LEADING COMPANIES USE BENCHMARKING -- STUDYING AND EMULATING TOP PERFORMERS INSIDE, AND OUTSIDE, THEIR INDUSTRIES -- TO ELIMINATE LONG-STANDING PROBLEMS AND BECOME TOP PERFORMERS.

DISCOVER HOW NEW PRACTICES CAN BE APPLIED TO YOUR ORGANIZATION -- WITH IMPRESSIVE AND MEASURABLE RESULTS -- IN BENCHMARKING, AN INNOVATIVE THREE-PART VIDEO SERIES. WE'LL TAKE YOU DEEP INSIDE PROFILED COMPANIES SUCH AS MOBIL OIL, GTE, AND SUNHEALTH TO LEARN HOW THEY IDENTIFIED "BEST OF CLASS" COMPANIES TO BENCHMARK IN ORDER TO IMPROVE THEIR OWN PERFORMANCE.

YOU'LL SEE HOW BENCHMARKING CAN GIVE YOUR TEAM A COMMON RALLYING POINT AND MOTIVATE COORDINATED ACTION. YOU'LL LEARN HOW TO IDENTIFY PROCESSES TO BENCHMARK, HOW TO FIND THE RIGHT PARTNER, AND HOW TO INITIATE THE FIRST STEPS (EVEN ON A LIMITED BUDGET). YOU'LL FIND OUT HOW TO IDENTIFY NOVEL OPPORTUNITIES, HOW TO STRUCTURE YOUR EFFORTS FOR SUCCESS, EVEN PROPER BENCHMARKING ETIQUETTE. EACH CONCEPT IS CLEARLY EXPLAINED AND ILLUSTRATED TO FACILITATE IMPLEMENTATION.

BENCHMARKING FOR CONTINUOUS IMPROVEMENT, BENCHMARKING CORE PROCESSES, AND BENCHMARKING OUTSIDE THE BOX BRING YOU FIRSTHAND COMMENTARY FROM SENIOR EXECUTIVES, INDUSTRY EXPERTS, AND FRONT-LINE PERSONNEL IN A FAST-PACED DOCUMENTARY STYLE THAT GENERATES INTEREST, UNDERSTANDING, AND ENTHUSIASM FOR THESE IMPORTANT IDEAS. THESE VIDEOS WILL STIMULATE DISCUSSION AND PROVIDE GUIDELINES TO HELP YOU DEVELOP AN ACTION PLAN FOR YOUR ORGANIZATION.

MAY I SEND YOU BENCHMARKING FOR A FREE, NO-OBLIGATION TRIAL? SIMPLY REPLY TO THIS E-MAIL WITH THE WORD "YES" IN THE SUBJECT LINE AND WE'LL SEND YOU THE PROGRAM TO TRY WITH OUR COMPLIMENTS. WE'LL SEND YOU THIS INNOVATIVE SERIES RIGHT AWAY. AFTER 14 DAYS, WE WILL MAIL YOU AN INVOICE FOR \$1190 (A SAVINGS OF \$595 VERSUS THE INDIVIDUAL VIDEO PRICE OF \$595 EACH).

IF YOU ARE NOT COMPLETELY SATISFIED WITH BENCHMARKING SIMPLY RETURN IT TO US. YOU WILL OWE NOTHING. WHY WAIT TO LEARN HOW SUCCESSFUL CHANGE MANAGEMENT CAN DRAMATICALLY ENHANCE YOUR ORGANIZATION'S PERFORMANCE?

SINCERELY,
LAURA WINIG

FIG. 2A

5/6

DIRECTOR

P.S. IF YOU PREFER, PRINT OUT THIS INVITATION, INITIAL IT AT THE TOP, (PLEASE VERIFY YOUR SHIPPING ADDRESS IS CORRECT AS LISTED ABOVE -- WE MUST HAVE A STREET ADDRESS FOR SHIPMENT) AND FAX IT TO 617-496-1029, OR SIMPLY CALL 1-800-668-6780. PLEASE BE SURE TO MENTION PRIORITY CODE 3275.

CONTACT INFORMATION

BELOW PLEASE FIND THE CONTACT INFORMATION WE CURRENTLY HAVE ON FILE. IF THIS INFORMATION IS NOT CORRECT, PLEASE MAKE YOUR EDITS BETWEEN THE APPROPRIATE BRACKETS AND RETURN - VERBATIM - AS PART OF YOUR REPLY E-MAIL. PLEASE INDICATE ANY ADDRESS CHANGE BY INCLUDING THE WORDS 'ADDRESS CHANGE' AT THE TOP OF YOUR ORDER-REPLY.

IF YOU WISH TO UNSUBSCRIBE FROM SPECIAL OFFER MAILINGS, PLEASE REPLY TO THIS E-MAIL MESSAGE WITH THE WORD "UNSUB" AT THE TOP OF YOUR REPLY.

BILLING ADDRESS

BILL FIRST NAME:	[WILLIAM]
BILL LAST NAME:	[HERP]
BILL TITLE:	[PRESIDENT]
BILL COMPANY:	[E-CARE GROUP INC.]
BILL DEPARTMENT:]	
BILL ADDRESS1:	[1646 MASSACHUSETTS AVE.]
BILL ADDRESS2:]
BILL ADDRESS3:]
BILL CITY:	[LEXINGTON]
BILL PROVINCE/STATE:	[MA]
BILL POSTAL/ZIP CODE:	[02173]
BILL COUNTRY:]
BILL PHONE:]
BILL FAX:]
BILL EMAIL:	[WHERP@E-CARE.COM]

SHIPPING ADDRESS (IF DIFFERENT)

SHIP FIRST NAME:]	}
SHIP LAST NAME:]	
SHIP TITLE:]	
SHIP COMPANY:]	
SHIP DEPARTMENT:]	
SHIP ADDRESS1:]	
SHIP ADDRESS2:]	
SHIP ADDRESS3:]	
SHIP CITY:]	
SHIP PROVINCE/STATE:]	
SHIP POSTAL/ZIP CODE:]	
SHIP COUNTRY:]	
SHIP PHONE:]	
SHIP FAX:]	
SHIP EMAIL:]	

[[891270||3275]]

68 70 72

FIG. 2B

6/6

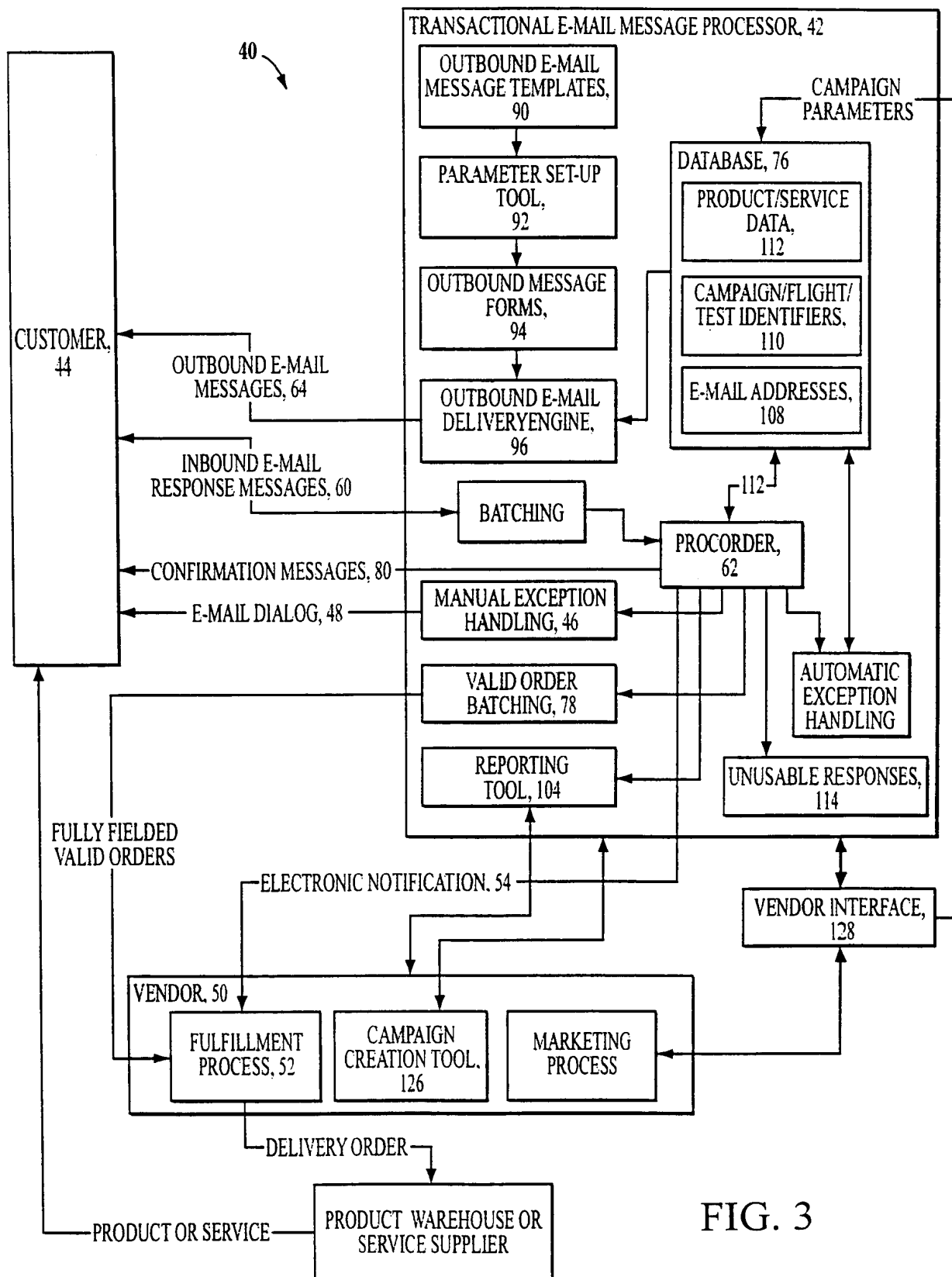


FIG. 3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/19403

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 17/60

US CL : 705/1

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 705/1,709/206,207,379/193,707/505

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

DIALOG, WEST,EAST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,724,424 A (GIFFORD) 03 Mar 1998, Fig 1[68,64,66,67,62,200]; col 4 lines 48-53; col 5 lines 22-29; Fig.3[5,6,7]; col 5 lines 18-19;col 5 lines 34-37;Fig 5[18]; Fig 4[15]; col 5 lines 49-57; Fig 6[19-25];col 5 lines 29-44;Fig 4[8-17];Fig 2[1-3];Fig 3[3];col 5 line 25-col 6 line 5];Fig. 5;col 6 lines 20-29;col 6 lines 39-42;Fig.3[3]	1-16



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:	*T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A" document defining the general state of the art which is not considered to be of particular relevance	*X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E" earlier document published on or after the international filing date	*Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	* & * document member of the same patent family
"O" document referring to an oral disclosure, use, exhibition or other means	
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search

19 AUGUST 2000

Date of mailing of the international search report

02 OCT 2000

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer

Tod Swann

Telephone No. (703) 305-3230

Regenia Logan